

**课 程 实 验 报 告**

**课程名称： C语言程序设计实验**

**专业班级： 本硕博202001班**

**学 号： U202112205**

**姓 名： 刘浩阳**

**指导教师： 吴俊军**

**报告日期： 2022年1月9日**

**网络空间安全学院**

**目 录**

[1表达式和标准输入输出实验 **1**](#_Toc404837920)

[1.1 实验目的 1](#_Toc404837921)

[1.2 实验内容 10](#_Toc404837922)

[1.3 实验小结 1](#_Toc404837923)0

[2 流程控制实验 **1**](#_Toc404837924)**0**

[2.1 实验目的 10](#_Toc404837921)

[2.2 实验内容 10](#_Toc404837922)

[2.3 实验小结 25](#_Toc404837923)

[3 函数与程序结构实验 **26**](#_Toc404837929)

[3.1 实验目的 26](#_Toc404837921)

[3.2 实验内容 26](#_Toc404837922)

[3.3 实验小结 42](#_Toc404837923)

[4 编译预处理实验 **43**](#_Toc404837934)

[4.1 实验目的 43](#_Toc404837921)

[4.2 实验内容 43](#_Toc404837922)

[4.3 实验小结 60](#_Toc404837923)

[5 数组实验 **61**](#_Toc404837938)

[5.1 实验目的 61](#_Toc404837921)

[5.2 实验内容 61](#_Toc404837922)

[5.3 实验小结 76](#_Toc404837923)

[6 指针实验 **77**](#_Toc404837943)

[6.1 实验目的 77](#_Toc404837921)

[6.2 实验内容 77](#_Toc404837922)

[6.3 实验小结 100](#_Toc404837923)

[7 结构与联合实验 **100**](#_Toc404837948)

[7.1 实验目的 100](#_Toc404837921)

[7.2 实验内容 100](#_Toc404837922)

[7.3 实验小结 118](#_Toc404837923)

[8 文件实验 **119**](#_Toc404837953)

[8.1 实验目的 119](#_Toc404837921)

[8.2 实验内容 119](#_Toc404837922)

[8.3 实验小结 127](#_Toc404837923)

[参考文献 **127**](#_Toc404837957)

# 实验1 表达式和标准输入与输出实验

## 1.1 实验目的

（1）熟练掌握各种运算符的运算功能，操作数的类型，运算结果的类型及运算过程中的类型转换，重点是C语言特有的运算符，例如位运算符，问号运算符，逗号运算符等；熟记运算符的优先级和结合性。

（2）掌握getchar, putchar, scanf 和printf 函数的用法。

（3）掌握简单C程序的编写方法。

（4）熟悉C语言程序的开发环境，并学会调试程序的方法。

## 1.2 实验内容

**1.2.1 程序改错与跟踪调试**

下面的实验1-1程序用来完成以下任务：

（1）输入华氏温度f，将它转换成摄氏温度c后输出。

（2）输入圆的半径值ｒ，计算并输出圆的面积ｓ。

（3）将ｋ的高字节作为结果的低字节，ｐ的高字节作为结果的高字节，拼成一个新的整数后输出。

在这个程序中存在若干语法和逻辑错误，要求先编译程序改正语法错误，再采用单步执行的方式调试程序找出逻辑错误。在单步执行程序的过程中，观察以下变量值：

（1）执行完c = 5/9 \* (f-32)，c的值为多少？

（2）执行完scanf(“%f”, &r)，r的值为多少？

（3）执行完newint = p&0xff00|k>>8，newint的值是多少？表达式k>>8的值是多少？

根据观察结果分析代码并修改程序，使之能够正确完成指定任务。

/\*实验1-1程序改错与跟踪调试题源程序\*/

1 #include<stdio.h>

2 #define PI 3.14159;

3 int main( void )

4 {

5 int f ;

6 short p, k ;

7 double c, r, s ;

/\* 任务1 \*/

8 printf("Input Fahrenheit: " ) ;

9 scanf("%d", f ) ;

10 c = 5/9 \* (f-32) ;

11 printf( "\n %d (F) = %.2f (C)\n\n ", f, c ) ;

/\* 任务2 \*/

12 printf("input the radius r:");

13 scanf("%f", &r);

14 s = PI \* r \* r;

15 printf("\nThe acreage is %.2f\n\n",&s);

/\* 任务3 \*/

16 k = 0xa1b2, p = 0x8432;

17 newint = p&0xff00|k>>8;

18 printf("newint = %#x\n\n",newint);

19 return 0;

20 }

**解答：**

（1）语法错误修改：

1) 第2行的符号常量定义后不能有分号，正确形式为：#define PI 3.14159

2) 第9行的scanf变量名前缺少&，正确形式为：scanf（“%d”，&f）；

3）第15行的printf变量名前多余&，正确形式为：

printf("\nThe acreage is %.2f\n\n",s);

4）第17行的newint未定义，应添加：unsigned short newint;

（2）修改完以上错误后，在单步执行过程中，观察以下变量值：

1. 执行完c = 5/9 \* (f-32)，c的值为：0.00。
2. 执行完scanf(“%f”, &r)，r的值为：0.000000。
3. 执行完newint = p&0xff00|k>>8，newint的值是0xffa1。

表达式k>>8的值是0xffffffa1。

（3）基于以上的单步执行观察结果，说明还存在如下逻辑错误：

1）c = 5/9 \* (f-32)； ，由于整型溢出，应该修改成：c = 5.0/9 \* (f-32)； 。

2）scanf(“%f”, &r)； ，由于%f是对应float类型，而r是double型，故应该修改成：scanf(“%lf”, &r)； 。

3）newint = p&0xff00|k>>8; ,由于k的补位会补1，所以只保留末8位，应该修改成：newint = p&0xff00|(k>>8&0x00ff); 。

修改后，源程序清单如下：

1 #include<stdio.h>

2 #define PI 3.14159

3 int main( void )

4 {

5 int f ;

6 short p, k ;

7 double c, r, s ;

/\* 任务1 \*/

8 printf("Input Fahrenheit: " ) ;

9 scanf("%d",& f ) ;

10 c = 5.0/9 \* (f-32) ;

11 printf( "\n %d (F) = %.2f (C)\n\n ", f, c ) ;

/\* 任务2 \*/

12 printf("input the radius r:");

13 scanf("%lf", &r);

14 s = PI \* r \* r;

15 printf("\nThe acreage is %.2f\n\n",s);

/\* 任务3 \*/

16 unsigned short newint;

17 k = 0xa1b2, p = 0x8432;

18 newint = p&0xff00|（k>>8&0x00ff）;

19 printf("newint = %#x\n\n",newint);

20 return 0;

21 }

（4）错误修改后运行结果如图1-1所示。

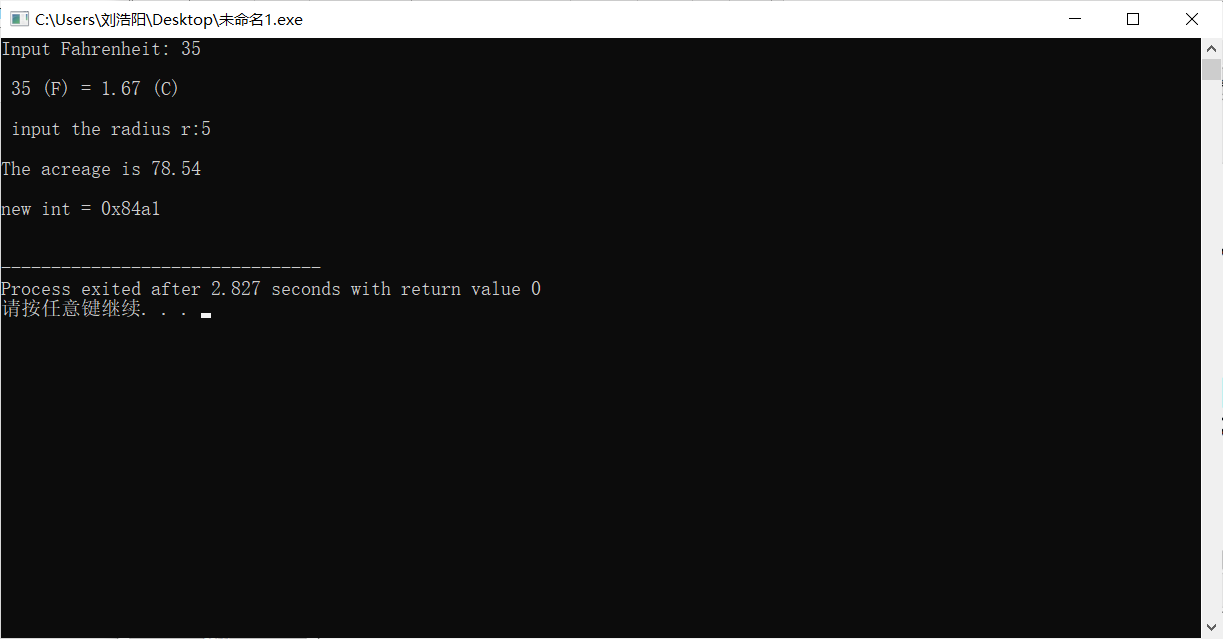


图1-1 实验1-1修改后运行结果截图

**1.2.2 程序设计**

（1）输入字符ｃ，如果ｃ是大写字母，则将ｃ转换成对应的小写，否则ｃ的值不变，输入Ctrl+Z程序结束。要求：①用条件表达式；②字符的输入输出用getchar和putchar函数。程序应能循环接受用户的输入，直至输入Ctrl+Z程结束。例如：

A （键盘输入）

a

^Z （键盘输入）

**解答：**

1. 算法流程如图1-2所示。



图1-2 程序设计题1的程序流程图

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | 测试  用例 | 程序输入 | 理论结果 | 运行结果 | | 用例1 | A | a | a | | 用例2 | c | c | c | | 用例3 | // | // | // | | 用例4 | 12 | 12 | 12 | | 用例5 | Ctrl+z | 结束 | 结束 | |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

2）源程序清单

#include<stdio.h>

int main()

{

char c;

while(c!=EOF)

{

c=getchar();

(c>='A'&&c<='Z')?c=c-'A'+'a':c;

if(c!=EOF)

putchar(c);

}

return 0;

}

3）测试

（a） 测试数据：如表1-1所示

表1-1 编程题1的测试数据

（b） 对应测试数据的运行结果截图

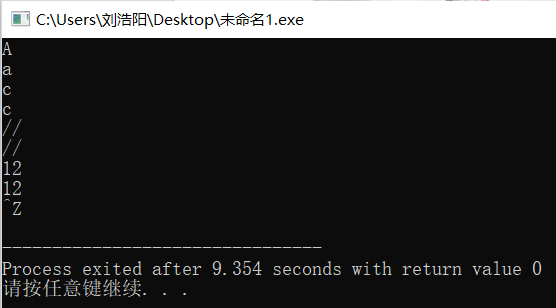


图1-3 程序设计题1的测试用例1、2、3、4、5的运行结果

（2）输入无符号短整数x，ｍ，ｎ（0 ≤ｍ≤ 15, 1 ≤ ｎ≤ 16-ｍ），取出x从第ｍ位开始向左的ｎ位（x从右至左编号为0～15），并使其向左端（第15位）靠齐。要求：①检查m和n的范围；②x的值以十六进制输入，m和n以十进制输入；③结果以十六进制输出。

**解答：**

1. 解题思路：如图1-4所示



图1-4程序设计题2的程序流程图

1. 程序清单：

#include<stdio.h>

int main()

{

unsigned short x,m,n;

scanf("%hx%hu%hu",&x,&m,&n);

if ((0<=m)&&(m<=15)&&(1<=n)&&(n<=(16-m)))

printf("%hx",x<<=(16-m-n));

else printf("error");

return 0;

}

3）测试

（a） 测试数据：如表1-2所示

表1-2 编程题2的测试数据

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 测试  用例 | 程 序 输 入 | | | 理 论 结 果 | 运 行 结 果 |
| X | m | n |
| 用例1 | 0010 1000 0001 0100（2814） | 5 | 6 | 280 | 280 |
| 用例2 | 1011 0000 1111 1010（b0fa） | 13 | 2 | 61f4 | 61f4 |
| 用例3 | 1111 0000 1100 0011（f0c3） | 13 | 5 | error | error |

（b） 对应测试测试用例一的运行结果如图1-5所示。

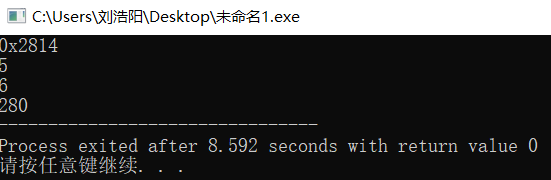


图1-5 程序设计题2的测试用例1的运行结果

对应测试测试用例二的运行结果如图1-6所示。

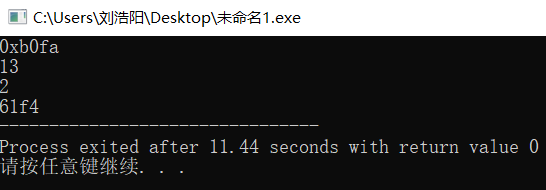


图1-6 程序设计题2的测试用例2的运行结果

对应测试测试用例三的运行结果如图1-7所示。

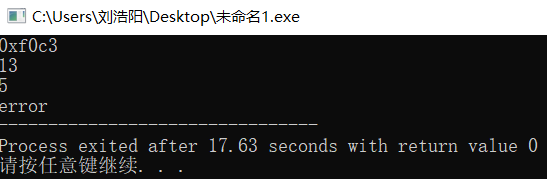


图1-7 程序设计题2的测试用例3的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

（3）读入无符号长整型数表示的互联网IP地址，对其译码，以常见的点分十进制形式输出。要求循环输入和输出，直至输入Ctrl+Z结束。

**解答**：

1. 解题思路：如图1-8所示



图1-8 程序设计题3的程序流程图

1. 程序清单

#include<stdio.h>

int main()

{ unsigned long IP,A,B,C,D;

const unsigned long a=0x000000ff;

const unsigned long b=0x0000ff00;

const unsigned long c=0x00ff0000;

const unsigned long d=0xff000000;

while(scanf("%lu",&IP)!=EOF){

A=a&IP;

B=b&IP;B>>=8;

C=c&IP;C>>=16;

D=d&IP;D>>=24;

printf("%lu.%lu.%lu.%lu\n",D,C,B,A);

}

return 0;

}

（a）测试数据如表1-3所示

表1-3 编程题3的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例1 | 3232235876 | 192.168.1.100 | 192.168.1.100 |
| 用例2 | 66322345376 | 113.30.173.160 | 113.30.173.160 |
| 用例3 | 1112223333 | 66.75.46.101 | 66.75.46.101 |
| 用例4 | 66322345376 1112223333 | 113.30.173.160 66.75.46.101 | 113.30.173.160 66.75.46.101 |
| 用例5 | Ctrl+z | 结束 | 结束 |

（b） 对应测试测试用例一的运行结果如图1-9所示。

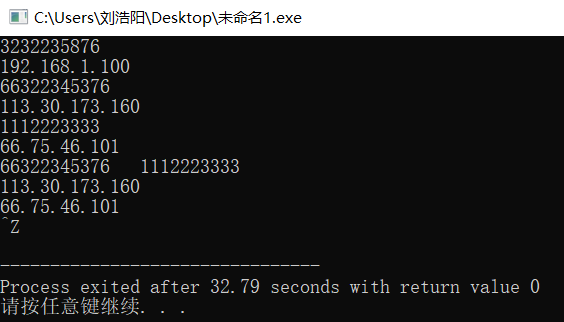


图1-9 程序设计题3的测试用例1的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

## 1.3 实验小结

由于本次是第一次实验，实验前对实验题目也没有过多的了解和尝试，导致实验时遇到了很多困难。其中遇到了一些不熟悉的循环语句的使用，比如while if等语句，这方面确实预习不够。还有一部分就是因为没有思路，比如实验1-3对mask的作用第一时间没有想到，导致刚开始做的时候毫无头绪，其中对EOF的使用方法也不是很清楚，后来在自己的查找书籍和与同学的讨论下才勉强解决。同时，在与同学的交流中，也更加熟悉了mask的作用，比如实验1-3中我定义了4个mask，有的同学只定义了1个mask同样也能实现目标，这对我简化代码提供了一定思路。这次实验，虽然没有在实验课上按时完成，但回去后也是很快解决了，是令我比较开心的，但同时也让我认识到了自己知识的不足与能力的欠缺，今后要增强对书本的预习复习包括自己的实践动手能力。

# 实验2 流程控制实验

## 2.1 实验目的

（1）掌握复合语句、if语句、switch语句的使用，熟练掌握for、while、do-while三种基本的循环控制语句的使用，掌握重复循环技术，了解转移语句与标号语句。

（2）练习循环结构for、while、do-while语句的使用。

（3）练习转移语句和标号语句的使用。

（4）使用集成开发环境中的调试功能：单步执行、设置断点、观察变量值。

## 2.2 实验内容及要求

**1．程序改错**

下面的实验2-1程序是合数判断器（合数指自然数中除了能被1和本身整除外，还能被其它数整除的数），在该源程序中存在若干语法和逻辑错误。要求对该程序进行调试修改，使之能够正确完成指定任务。

/\* 实验2-1改错题程序：合数判断器\*/

1 #include <stdio.h>

2 int main( )

3 {

4 int i, x, k, flag = 0;

5 printf("本程序判断合数，请输入大于1的整数，以Ctrl+Z结束\n");

6 while (scanf("%d", &x) !=EOF) {

7 for(i=2,k=x>>1;i<=k;i++)

8 if (!x%i) {

9 flag = 1;

10 break;

11 }

12 if(flag=1) printf("%d是合数", x);

13 else printf("%d不是合数", x);

14 }

15 return 0;

16 }

**解答：**

1. 语法错误修改：
2. 12行if判断中flag=1应为判断语句，正确形式为：flag==1。
3. 14行添加flag=0重置flag值。
4. 修改完以上错误，在单步执行过程中，观察以下变量值：
5. 执行完for(i=2,k=x>>1;i<=k;i++),flag的值总为1。

1. 基于以上的单步执行观察结果，说明存在如下逻辑错误：
2. if(!x%i)，由于运算符优先级!先于%，应该修改成：if(!(x%i))

修改后，源程序清单如下：

#include <stdio.h>

int main( )

{

int i, x, k, flag = 0;

printf("本程序判断合数，请输入大于1的整数，以Ctrl+Z结束\n");

while (scanf("%d", &x) !=EOF) {

flag=0;

for(i=2,k=x>>1;i<=k;i++)

if (!(x%i)) {

flag = 1;

break;

}

if(flag==1) printf("%d是合数\n", x);

else printf("%d不是合数\n", x);

flag=0;

}

return 0;

}

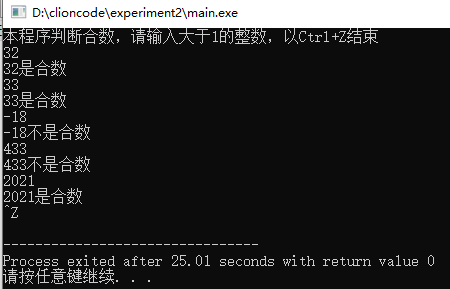
1. 错误修改后运行结果如图2-1所示。

图2-1 实验2-1修改后运行结果截图

**2．程序修改替换**

（1）修改实验2-1程序，将内层两出口的for循环结构改用单出口结构，即不允许使用break、goto等非结构化语句。

**解答：**

1. 解题思路：

I．删除break，则for循环只有一个输出出口。

1. 源程序清单

#include <stdio.h>

int main( )

{

int i, x, k, flag = 0;

printf("本程序判断合数，请输入大于1的整数，以Ctrl+Z结束\n");

while (scanf("%d", &x) !=EOF) {

flag=0;

for(i=2,k=x>>1;i<=k;i++)

if (!(x%i)) {

flag = 1;

}

if(flag==1) printf("%d是合数\n", x);

else printf("%d不是合数\n", x);

}

return 0;

}

1. 测试

（a）测试数据：如表2-1所示

表2-1 程序修改题1的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例1 | 4 | 4是合数 | 4是合数 |
| 用例2 | 34 | 34是合数 | 34是合数 |
| 用例3 | -66 | -66不是合数 | -66不是合数 |
| 用例4 | 2021 | 2021是合数 | 2021是合数 |
| 用例5 | 0 | 0不是合数 | 0不是合数 |
| 用例6 | Ctrl+z | 结束 | 结束 |

（b）对应测试数据的运行结果如图2-2所示

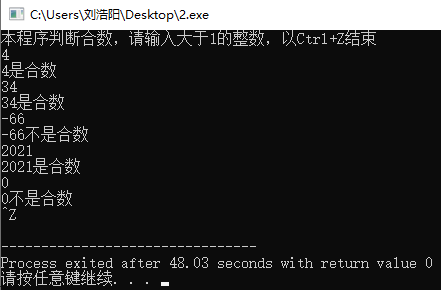


图2-2 程序修改题1的测试用例1、2、3、4、5、6的运行结果

（2）修改实验2-1程序，将for循环改用do-while循环。

**解答：**

1. 解题思路：
2. 输入x。
3. 判断是否输入Ctrl+z。是则结束程序，不是则执行循环语句。
4. 循环语句中判断x%i是否等于0且x是否大于1，是则改变flag值为1，结束do-while循环。否则将i值加1，再次执行循环。
5. do-while循环后判断flag值，若等于1则输出“x是合数”，否则输出“x不是合数”，并重新执行while循环。
6. 程序清单

#include <stdio.h>

int main() {

int i,x,k,flag;

printf("本程序判断合数，请输入大于1的整数，以Ctrl+z结束\n");

while(scanf("%d",&x)!=EOF){

i=2;

do

{

k=x>>1;

flag=0;

if ((!(x%i))&&(x>1)) {flag=1;break;}

else ++i;

}

while (i<=k);

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例1 | 23 | 23不是合数 | 23不是合数 |
| 用例2 | 34 | 34是合数 | 34是合数 |
| 用例3 | 0 | 0不是合数 | 0不是合数 |
| 用例4 | -111 | -111不是合数 | -111不是合数 |
| 用例5 | Ctrl+z | 结束 | 结束 |

if(flag==1) printf("%d是合数",x);

else printf("%d不是合数",x);

}

return 0;

}

1. 测试
2. 测试数据：如表2-2所示

表2-2 程序修改题2的测试数据

b）对应测试的运行结果如图2-3所示。

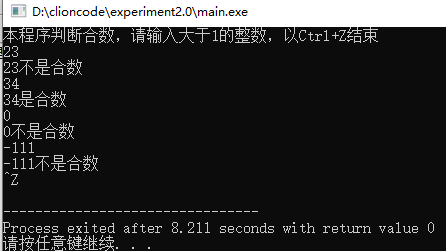


图2-3 程序修改题2的测试用例1、2、3、4、5的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

（3）修改实验2-1程序，将其改为纯粹合数求解器，求出所有的3位纯粹合数。一个合数去掉最低位，剩下的数仍是合数；再去掉剩下的数的最低位，余留下来的数还是合数，这样反复，一直到最后剩下一位数仍是合数，这样的数称为纯粹合数。

**解答：**

1. 解题思路：
2. 输入x=100。
3. 定义变量k，i。分别是x去掉个位数和去掉个十位的数。
4. 判断x，k，i是否为合数，若同时为合数，则输出x，否则使x值加1，重新执行循环，直至x=999。
5. 程序清单：

#include <stdio.h>

int main()

{

int x,k,i,m;

for(x=100;x<=999;x++)

{

k=x/10;

i=k/10;

for(m=2;m<i;m++)

{

if ((i % m == 0) && (k % m == 0) && (x % m == 0))

printf("%d ", x);

}

}

return 0;

}

1. 输出结果如图2-4所示

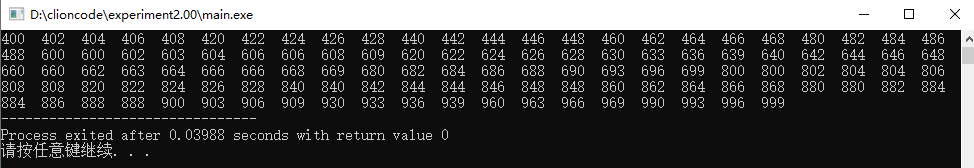


图2-4 程序修改题3的运行结果

**3．程序设计**

（1） 假设工资税金按以下方法计算：x ＜ 1000元，不收取税金；1000 ≤ x ＜ 2000，收取5%的税金；2000 ≤ x ＜ 3000，收取10%的税金；3000 ≤ x ＜ 4000，收取15%的税金；4000 ≤ x ＜ 5000，收取20%的税金；5000 ≤ x，收取25%的税金。（注意税金的计算按照阶梯计税法，比如，工资为4500，那么税金=1000\*5% + 1000\*10% + 1000\*15% + 501\*20%）。编写一个程序,输入工资金额，输出应收取税金额度，要求分别用if语句和switch语句来实现。

**解答：**

1. 解题思路：
2. 输入salary
3. 取salary的千位数，并利用switch或if语句，将salary带入不同的tax计算公式，得出tax的值并输出。

2）程序清单：

If语句：

#include <stdio.h>

int main()

{

double salary,tax;

printf("请输入工资：\n");

scanf("应交税：%lf",&salary);

if (salary<1000)

printf("应交税：0");

else if((salary>=1000)&&(salary<2000))

printf("应交税：%lf ",tax=(salary-1000)\*(0.05));

else if((salary>=2000)&&(salary<3000))

printf("应交税：%lf ",tax=((salary-2000)\*(0.1)+50));

else if((salary>=3000)&&(salary<4000))

printf("应交税：%lf ",tax=(salary-3000)\*(0.15)+150);

else if((salary>=4000)&&(salary<5000))

printf("应交税：%lf ",tax=(salary-4000)\*(0.2)+300);

else if(salary>=5000)

printf("应交税：%lf ",tax=(salary-5000)\*(0.25)+500);

return 0;

}

Switch语句：

#include <stdio.h>

int main()

{

double salary,tax,x;

printf("请输入工资：\n");

scanf("%lf",&salary);

x=salary/1000;

if(x>1) {

switch ((int)x) {

case 1:

printf("应交税：%lf", tax = (salary - 1000) \* (0.05));break;

case 2:

printf("应交税：%lf", tax = (salary - 2000) \* (0.1) + 50);break;

case 3:

printf("应交税：%lf", tax = (salary - 3000) \* (0.15) + 150);break;

case 4:

printf("应交税：%lf", tax = (salary - 4000) \* (0.2) + 300);break;

default:

printf("应交税：%lf", tax = (salary - 5000) \* (0.25) + 500);break;

}

}

else printf("应交税：0");

return 0;

}

3）测试

1. 测试数据如表2-3所示：

表2-3 编程题1的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例1 | 999 | 0 | 0 |
| 用例2 | 1800 | 40.000000 | 40.000000 |
| 用例3 | 2700 | 120.000000 | 120.000000 |
| 用例4 | 3600 | 240.000000 | 240.000000 |
| 用例5 | 4500 | 400.000000 | 400.000000 |
| 用例6 | 5400 | 600.000000 | 600.000000 |

1. 对应测试用例的运行结果如下图所示

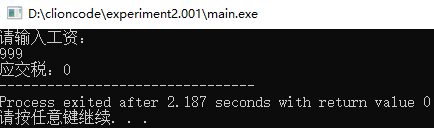


图2-5 程序设计题1的测试用例1的运行结果

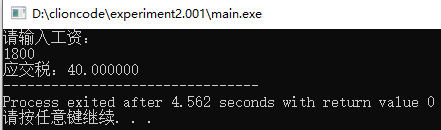


图2-6 程序设计题1的测试用例2的运行结果

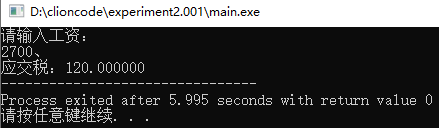


图2-7 程序设计题1的测试用例3的运行结果

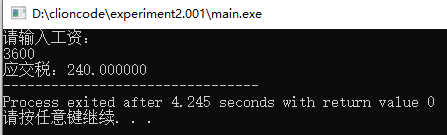


图2-8 程序设计题1的测试用例4的运行结果

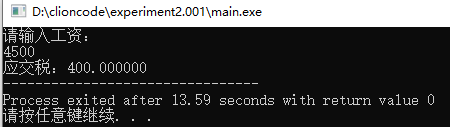


图2-9程序设计题1的测试用例5的运行结果

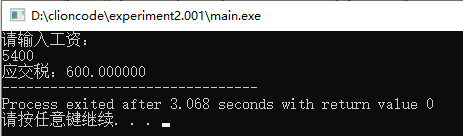


图2-10 程序设计题1的测试用例6的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

（2）输入一段以!结尾的短文(最多5行,每行不超过50个字符)，要求将它复制到输出，复制过程中将每行一个以上的空格字符用一个空格代替。

**解答：**

1. 解题思路：如图2-11所示



图2-11 程序设计题2的程序流程图

1. 程序清单

#include <stdio.h>

int main()

{

char c;

enum{A,B,C};

int state=A;

while('!'!=(c=getchar()))

{

switch (state)

{

case 0:

if (c == ' ')

{putchar(c), state = B;}

else

{putchar(c);}

break;

case 1:

if (c != ' ') { putchar(c), state = A; }

break;

}

}

putchar('!');

return 0;

}

1. 测试
2. 测试数据如表2-4所示

表2-4 编程题2的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例1 | ABCD DCBA！ | ABCD DCBA！ | ABCD DCBA！ |
| 用例2 | C语言 实验！ | C语言 实验！ | C语言 实验！ |
| 用例3 | 123 456! | 123 456! | 123 456! |

1. 对应测试的运行结果如图2-12所示。

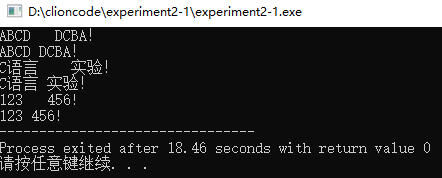


图2-12 程序设计题2的测试用例1、2、3的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

（3）打印杨辉三角，根据公式，用顺推法编程，输入最后一行的编号N（0<=N<=6），要求输出金字塔效果的杨辉三角形。特别要注意空格的数目，每个数字占位4格，例如，1位数后补3个空格。第N行行首是N个空格（每向上一行，行首空格数对应增加）。每行末尾是换行符。

**解答：**

1）解题思路：

输入行数N，定义空格数h，输出的数a，b，循环变量i，j。每行每行打印，利用for循环嵌套，每行先打出第一个数1以及1前面的空格，在根据递推公式打印1后面的数，并同时进行位数判断，再根据位数打印相应的空格数，每行打印完都换行，并使i值加1。如此循环II步骤直至循环变量i=N，停止循环，输出结果。

2）程序清单：

#include<stdio.h>

int main()

{

int a=1;

int i,j,b,c,N,h;

scanf("%d",&N);

for(i=0;i<=N;)

{

for(h=1;h<=N+2\*(N-i);h++)

printf(" ");

printf("%d",a);

printf(" ");

for(j=1;j<=i;)

{

b=a\*(i-j+1)/j;

printf("%d",b);

c=b/10;

if (c!=0)printf(" ");

else printf(" ");

a=b;j++;

}

printf("\n");i++;

}

return 0;

}

3）测试：

选取N=0 N=4 N=6 三个数据测试，结果如下图

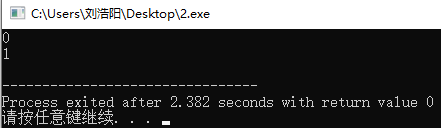


图2-13 程序设计题3的运行结果

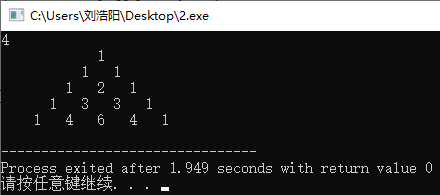


图2-14 程序设计题3的运行结果

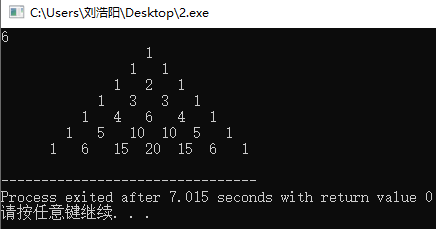


图2-15 程序设计题3的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

（4）625这个数很特别，625的平方等于390625，其末3位也是625。请编程输出所有这样的3位数：它的平方的末3位是这个数本身。要求这些数字从小到大排列，每个数字单独占一行。

**解答：**

1. 解题思路：如图2-16所示
2. 程序清单：

#include<stdio.h>

int main() {

int a=100;

while((100<=a)&&(a<=999))

{

if(a==((a\*a)%1000))

printf("%d\n",a),a+=1;

else a+=1;

}

return 0;

}



图2-16 程序设计题4的程序流程图

3）测试结果如图2-17所示

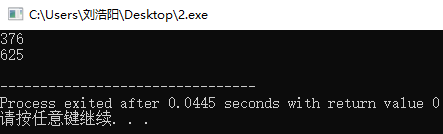


图2-17 程序设计题4的运行结果

## 2.3 实验小结

这次实验遇上的最大的问题就是打印杨辉三角的问题，其实主要还是表现在对for循环的使用不够熟练和顺手，遇到2-3个for循环嵌套就乱了套了，包括自己对题目的理解与解读，对一些算法没能足够的精简与优化。即使到最后，杨辉三角那题也是依靠同学的帮助才完成的，虽然我理解了同学们写的程序，但自己上手的时候还是不够完美的完成。希望以后多多练习这种题，这样才能提高自己的能力。

# 实验3 函数与程序结构实验

## 3.1实验目的

（1）熟悉和掌握函数的定义、声明；函数调用与参数传递，函数返回值类型的定义和返回值使用。

（2）熟悉和掌握不同存储类型变量的使用。

（3）练习使用集成开发环境中的调试功能：单步执行、设置断点、观察变量值。

## 3.2实验内容

**1．程序改错题**

下面是计算s=1!+2!+3!+…+n!的源程序(n<20)。在这个源程序中存在若干语法和逻辑错误。要求对该程序进行调试修改，使之能够输出如下结果：

k=1 the sum is 1

k=2 the sum is 3

k=3 the sum is 9

……

k=20 the sum is 2561327494111820313

/\*实验3-1改错题程序：计算s=1!+2!+3!+…+n!\*/

1. #include <stdio.h>
2. int main(void)
3. {
4. int k;
5. for(k=1;k<=20;k++)
6. printf("k=%d\tthe sum is %ld\n",k,sum\_fac(k));
7. return 0;
8. }
9. long sum\_fac(int n)
10. {
11. long s=0;
12. int i,fac;
13. for(i=1;i<=n;i++)
14. fac\*=i;
15. s+=fac;
16. return s;
17. }

**解答：**

1. 错误修改：
2. 对于大数的阶乘和，会造成数据溢出，应将函数类型以及函数内变量类型改为long long类型，long long sum\_fac；long long fac，s；
3. 未提前声明函数类型，在main函数前添加：long long sum\_fac；
4. 修改后的程序清单如下：

#include <stdio.h>

long long sum\_fac(int n);

int main(void)

{

int k;

for(k=1;k<=20;k++)

printf("k=%d\tthe sum is %llu\n",k,sum\_fac(k));

return 0;

}

long long sum\_fac(int n)

{

long long s=0,i,fac=1;

for(i=1;i<=n;i++)

{

fac\*=i;

s+=fac;

}

return s;

}

1. 修改后的运行结果如图3-1所示：

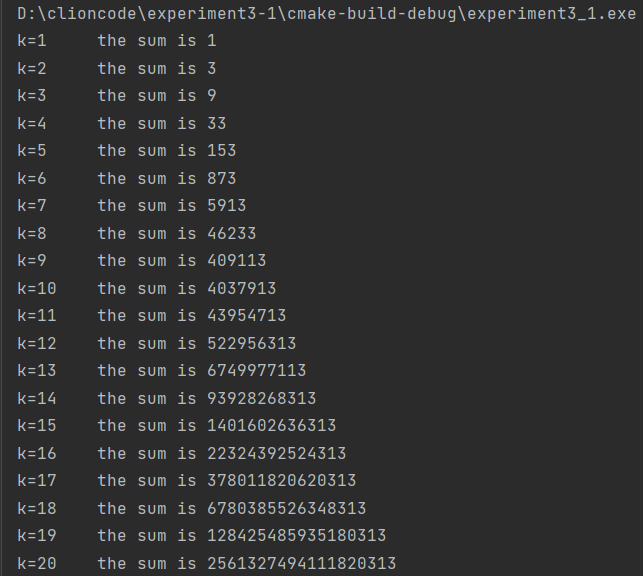


图3-1 程序改错题修改后运行结果截图

**2．程序修改替换题**

（1）根据将实验3-1改错题程序中sum\_fac函数修改为一个递归函数，用递归的方式计算。

**解答：**

1. 解题思路：

由递归思想，设计一个plus函数，用于进行对n！的计算以及阶乘的相加

1. 程序清单如下：

#include "stdio.h"

long long plus(int n){

int i=1;

long long fac=1;

if(n==1) return 1;

else{

for(i=1;i<=n;i++)

fac\*=i;

return fac+ plus(n-1);

}

}

int main(){

int n;

for(n=1;n<=20;n++)

printf("k=%d\tthe sum is %lld\n",n, plus(n));

return 0;

}

1. 运行结果如图3-2所示：

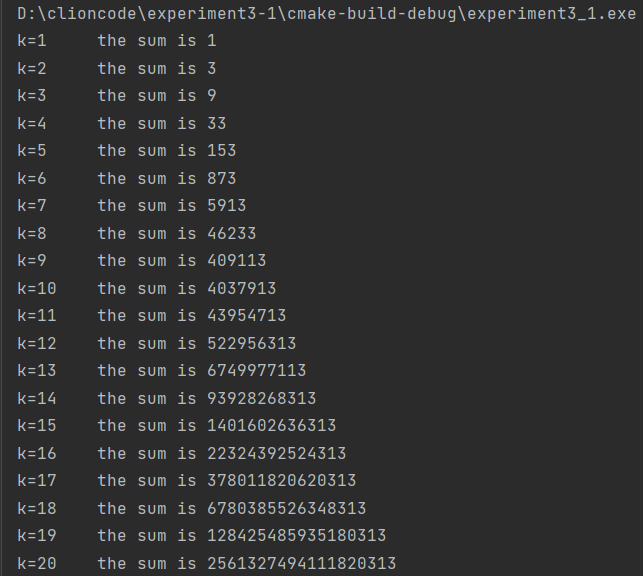


图3-2 程序修改题1运行结果截图

说明上述的运行结果与原运行结果吻合，验证了程序的正确性。

（2）下面是计算的源程序，其中x是浮点数，n是整数。从键盘输入x和n，然后计算s的值。修改该程序中的sum和fac函数，使之计算量最小。

/\*实验3-2程序修改替换第(2)题程序：根据公式计算 s\*/

#include<stdio.h>

double mulx(double x,int n);

long fac(int n);

double sum(double x,int n)

{

int i;

double z=1.0;

for(i=1;i<=n;i++)

{

z=z+mulx(x,i)/fac(i);

}

return z;

}

double mulx(double x,int n)

{

int i;

double z=1.0;

for(i=0;i<n;i++)

{

z=z\*x;

}

return z;

}

long fac(int n)

{

int i;

long h=1;

for(i=2;i<=n;i++)

{

h=h\*i;

}

return h;

}

int main()

{

double x;

int n;

printf("Input x and n:");

scanf("%lf%d",&x,&n);

printf("The result is %lf:",sum(x,n));

return 0;

}

**解答：**

1. 解题思路：

要使运算次数最少，则应将每一次的计算结果保存下来，将sum函数中z变量定义为static类型，并将return改为return z\*=x，同理对fac函数进行同样的处理即可。

（2） 程序清单：

#include<stdio.h>

#include<time.h>

double mulx(double x,int n);

long fac(int n);

double sum(double x,int n)

{

int i;

double z=1.0;

for(i=1;i<=n;i++)

{

z=z+mulx(x,i)/fac(i);

}

return z;

}

double mulx(double x,int n)

{

static double z=1.0;

return z\*=x;

}

long fac(int n)

{

static long h=1;

return h\*=n;

}

int main()

{

double x;

int n;

printf("Input x and n:");

scanf("%lf%d",&x,&n);

printf("The result is %lf\n",sum(x,n));

return 0;

}

1. 测试：
2. 测试数据如表3-1所示

表3-1 程序修改替换题2的测试数据

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试用例 | 程序输入 | | 理论结果 | 运行结果 |
| x | n |
| 用例1 | 3 | 3 | 13.000000 | 13.000000 |
| 用例2 | 14 | 7 | 38025.933333 | 38025.933333 |
| 用例3 | -1 | 5 | 0.366667 | 0.366667 |

1. 程序运行结果如下图所示：

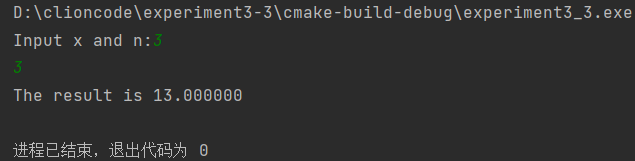


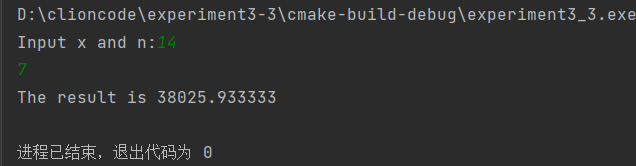
图3-3 程序修改替换题2运行结果截图

图3-4 程序修改替换题2运行结果截图

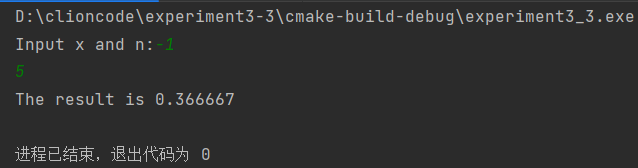


图3-5 程序修改替换题2运行结果截图

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

**3．跟踪调试题**

下面是计算fabonacci数列前n项和的源程序，现要求单步执行该程序，在watch窗口中观察Ik,sum,n值。具体操作如下：

（1）设输入5，观察刚执行完“scanf("%d",&k);”语句时，sum、k的值是多少？

（2）在从main函数第一次进入fabonacci函数前的一刻，观察各变量的值是多少？返回后光条停留在哪个语句上？

（3）在从main函数第一次进入fabonacci函数后的一刻，观察光条从main函数“sum+=fabonacci(i);”语句调到了哪里？

（4）在fabonacci函数内部单步执行，观察函数的递归执行过程。体会递归方式实现的计算过程是如何完成数计算的，并特别注意什么时刻结束递归，然后直接从第一个return语句返回到了哪里？

（5）在fabonacci函数递归执行过程中观察参数n的变化情况，并回答为什么k、sum在fabonacci函数内部不可见？

**/\*实验3-3跟踪调试题程序：**计算fabonacci数列前n项和**\*/**

#include<stdio.h>

int main(void)

{

int i,k;

long sum=0,fabonacci(int n);

printf("Inut n:");

scanf("%d",&k);

for(i=1;i<=k;i++){

sum+=fabonacci(i);

printf("i=%d\tthe sum is %ld\n",i,sum);

}

return 0;

}

long fabonacci(int n)

{

if(n==1 || n==2)

return 1;

else

return fabonacci(n-1)+fabonacci(n-2);

}

**解答：**

1. 输入5后，执行完“scanf("%d",&k);”语句时，sum=0，k=5。如图3-6所示

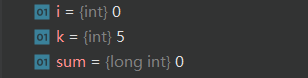


图3-6 跟踪调试题观察结果（1）截图

1. 在从main函数第一次进入fabonacci函数前的一刻，i=1，k=5，sum=0。如图3-7所示。返回后光条停留在“printf("i=%d\tthe sum is %ld\n",i,sum);”语句上。如图3-8所示。

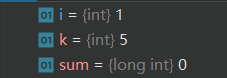


图3-7 跟踪调试题观察结果（2）截图

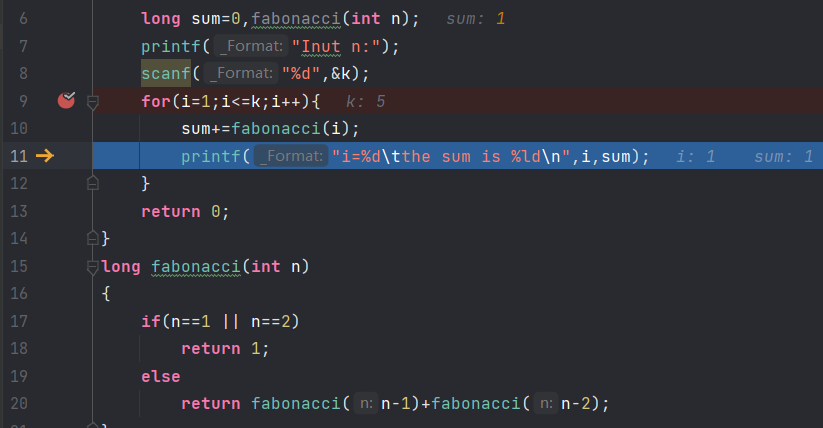


图3-8 跟踪调试题观察结果（2）截图

1. 在从main函数第一次进入fabonacci函数后的一刻，光条从main函数“sum+=fabonacci(i);”语句调到了“if(n==1 || n==2);”语句。如图3-9所示。

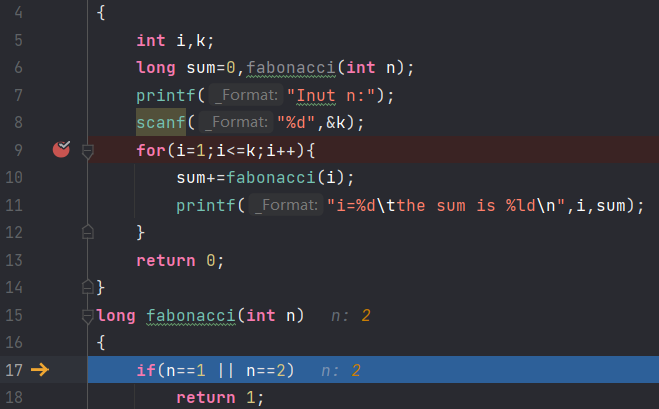


图3-9 跟踪调试题观察结果（3）截图

1. 在fabonacci函数中单步执行，发现递归过程中fabonacci函数先接收输入的i值，再进行判断i的值，若为1或2，则函数的返回值为1；若不为1或2，则返回fabonacci(n-1)+fabonacci(n-2)的值并进入fabonacci(n-1)函数，自此每次执行函数，直至i的值为1或2结束递归，得到返回值1，再重新进行累加，得到fabonacci(i)的值。然后直接从第一个return语句返回到“printf("i=%d\tthe sum is %ld\n",i,sum);”语句。
2. 若参数n值为1或2，则不变。若大于2，则每次递归都减1，直至值为1，最后返回时值变为原值。由于k，sum变量类型为auto，作用域为main函数块，故无法观察到k，sum值。

**4．程序设计题**

（1）编程验证歌德巴赫猜想：任何一个大于等于4的偶数都是两个素数之和。要求设计一个函数，接受形参n，以“n=n1+n2”的形式输出结果，若有多种分解情况，取n1最小的一个输出。

例如：n=6，输出“6=3+3”并换行。

main函数循环接收从键盘输入的整数n，如果n是大于或等于4的偶数，调用上述函数进行验证，直至输入Ctrl+Z程序结束。

**解答：**

1. 解题思路：如图3-10所示



图3-10 程序设计题1的程序流程图

1. 程序清单：

#include <stdio.h>

#include "math.h"

int prime(int x){

for(int i=2;i<=sqrt(x);i++)

{

if(!(x%i)) return 0;

}

return 1;

}

int p(int x){

int a,b;

for (a = 2; a < x; a++)

{

b = x - a;

if (prime(a) && prime(b))

{

printf("%d=%d+%d\n", x, a, b);

break;

}

}

}

int main()

{

int n,a,b;

while ((scanf("%d",&n))!=EOF)

{

if(n>=4) {

p(n);

}

else break;

}

return 0;

}

1. 测试：
2. 测试数据如表3-2所示

表3-2 程序设计题1的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例1 | 6 | 6=3+3 | 6=3+3 |
| 用例2 | 88 | 88=5+83 | 88=5+83 |
| 用例3 | -1 | 结束 | 结束 |
| 用例4 | Ctrl+z | 结束 | 结束 |

1. 测试结果如下图所示：

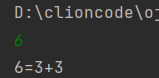


图3-11 程序设计题1的用例1的运行结果

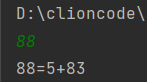


图3-12 程序设计题1的用例2的运行结果

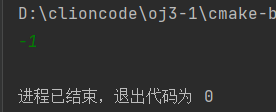


图3-13 程序设计题1的用例3的运行结果

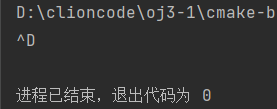


图3-14 程序设计题1的用例4的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

（2）完全数（Perfect number），又称完美数或完备数，特点是它的所有真因子（即除了自身以外的约数，包括1）之和恰好等于它本身。例如6=1+2+3，28=1+2+4+7+14等。

编程寻找10000以内的所有完全数。

要求设计一个函数，判定形参n是否为完全数，如果是，返回1，否则返回0。在main函数中调用该函数求10000以内的所有完全数，并以完全数的真因子之和的形式输出结果，例如“6=1+2+3”。程序输出中，每个完全数单独占一行。

**解答：**

1. 解题思路：如图3-15所示。



图3-15 程序设计题2的程序流程图

1. 程序清单：

#include <stdio.h>

int find(int n)

{

int term=0;

for(int i=1;i<n;i++)

{

if (n % i == 0) term += i;

}

if(term==n) return 1;

else return 0;

}

int main() {

int n,term=0;

for(n=3;n<10000;n++)

{

if(find(n))

{

printf("%d=1+",n);

for(int i=2;i<n;i++)

{

if(n%i==0)

{

term+=i;

if(term==n-1) printf("%d\n",i);

else printf("%d+",i);

}

}

term=0;

}

}

return 0;

}

1. 测试：运行结果如图3-16所示

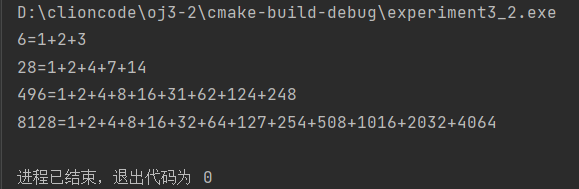


图3-16 程序设计题2的运行结果

（3）自幂数是指一个n位数，它每位数字的n次幂之和等于它本身。

水仙花数是3位的自幂数，除此之外，还有4位的四叶玫瑰数、5位的五角星数、6位的六合数、7位的北斗星数、8位的八仙数等。

编写一个函数，判断其参数n是否为自幂数，如果是，返回1；否则，返回0。

要求main函数能反复接收从键盘输入的整数k，k代表位数，然后调用上述函数求所有k位的自幂数，输出相应信息并换行，例如“3位的水仙花数共有4个153,370,371,407\n”。当k=0时程序结束执行。

**解答：**

1. 解题思路：
2. 设计函数判断一个数是否为自幂数。
3. 用while循环不断接收输入的k值，判断k的值，若为0，则结束程序，否则进入for循环，最小一个数开始判断是否为自幂数，同时用变量count记录个数。
4. 利用switch函数打印汉字前缀，同时一个一个打印自幂数。
5. 当打印到倒数第二个自幂数时，跳出for循环，进入下一个for循环，并打印最后一个自幂数并换行。
6. 程序清单：

#include <stdio.h>  
#include "math.h"  
int find(int n){  
    int x=0,z=0,i=0,term=0,t;  
    t=n;  
    z=n/10;  
    for(i=1;z>0;i++)  
        z/=10;//i是位数  
    x=n%10;  
    for(int j=i;j>=1;j--)  
    {  
        term+= pow(x,i);  
        n/=10;  
        x=n%10;  
    }  
    if(term==t) return 1;  
    else return 0;  
}  
int main()  
{  
    int k,count=0,n=1,t=0;  
    while(scanf("%d",&k)!=EOF)  
    {  
        count=0;  
        if(k==0) break;  
        else  
        {  
            for (int i = pow(10, k - 1); i < pow(10, k); i++)  
                if (find(i)) count++;  
            switch (k)  
            {  
                case 3:  
                    printf("3位的水仙花数共有%d个", count);break;  
                case 4:  
                    printf("4位的四叶玫瑰数共有%d个", count);break;  
                case 5:  
                    printf("5位的五角星数共有%d个", count);break;  
                case 6:  
                    printf("6位的六合数共有%d个", count);break;  
                case 7:  
                    printf("7位的北斗星数共有%d个", count);break;  
                case 8:  
                    printf("8位的八仙数共有%d个", count);break;  
            }  
            if(count==1) {  
                for (int i = pow(10, k - 1); (i < pow(10, k)); i++)  
                    if(find(i)) printf("%d\n",i);  
            }  
            else{  
                n=1;  
                for (int i = pow(10, k - 1); (i < pow(10, k))&&(n<count); i++)  
                {  
                    if(find(i)) {printf("%d,",i);n++;}  
                    t=i;  
                }  
                for(int z=t+1;z< pow(10,k);z++)  
                    if(find(z)) printf("%d\n",z);  
            }  
        }  
    }  
    return 0;  
}

1. 测试：
2. 测试数据如表3-3所示：

表3-3 程序设计题3的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 |
| 用例1 | 0 | 结束 |
| 用例2 | 5 | 5位的五角星数共有3个54748,92727,93084 |
| 用例3 | 3 5 7 0 | 3位的水仙花数共有4个153,370,371,407  5位的五角星数共有3个54748,92727,93084  7位的北斗星数共有4个1741725,4210818,9800817,9926315  结束 |

1. 运行结果如下图所示：

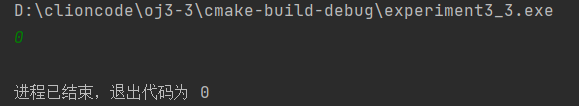


图3-17程序设计题3的测试用例1的运行截图

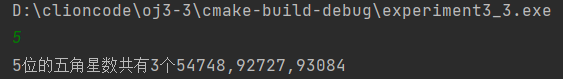


图3-18程序设计题3的用例2的运行截图

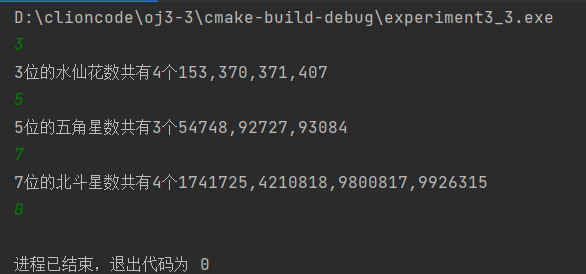


图3-19序设计题3的用例3的运行截图

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

## 3.3 实验小结

本次实验中，遇见了static变量类型的使用，以及学习了一点time函数的运用。其中最难的是打印自幂数的题，其实实验的思路并不难，但是想要将思路转换为代码还有点困难，其中我也发现了一个容易被忽视的小细节，就是循环语句中，每次循环后对变量的重置，我就是在这个点上卡了很久，之后才发现是while循环中没有对计数变量重新赋值，导致重复使用while循环语句的错误。

# 实验4 编译预处理实验

## 4.1 实验目的

（1）掌握文件包含、宏定义、条件编译和assert宏的使用；

（2）练习使用集成开发环境中的调试功能：单步执行、设置断点、观察变量值。

（3）熟悉多文件编译技术

## 4.2实验内容

**1．程序改错**

下面是用宏来计算平方差、交换两数的源程序.在这个源程序中存在若干错误，要求对该程序进行调试修改，使之能够正确完成指定任务。

/\*实验4-1改错与跟踪调试题程序：计算平方差、将换两数\*/

1 #include<stdio.h>

2 #define SUM a+b

3 #define DIF a-b

4 #define SWAP(a,b) a=b,b=a

5 int main()

6 {

7 int a,b;

8 printf("Input two integers a, b:");

9 scanf("%d%d", &a,&b);

10 printf("\nSUM=%d\n the difference between square of a and square of b is:%d",SUM, SUM\*DIF);

11 SWAP(a,b);

12 printf("\nNow a=%d,b=%d\n",a,b);

13 return 0;

14 }

**解答：**

1. 错误修改
2. 第2行a+b应改为（a+b）。
3. 第3行a-b应改为（a-b）。
4. 第4行应改为：#define SWAP（a，b，t） （t=a，a=b，b=t）
5. 第7行应添加t的定义：int t；
6. 第11行改为SWAP（a，b，t）
7. 修改后的程序为：

#include<stdio.h>

#define SUM (a+b)

#define DIF (a-b)

#define SWAP(a,b,t) (t=a,a=b,b=t)

int main()

{

int a,b,t;

printf("Input two integers a, b:");

scanf("%d%d", &a,&b);

printf("\nSUM=%d\n the difference between square of a and square of b is:%d",SUM, SUM\*DIF);

SWAP(a,b,t);

printf("\nNow a=%d,b=%d\n",a,b);

return 0;

}

1. 测试：

当输入0 0时，测试结果如图4-1所示

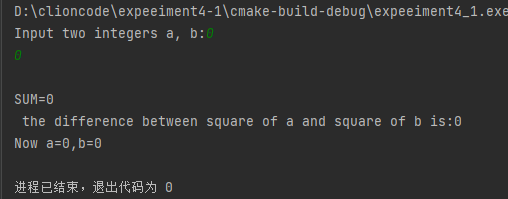


图4-1 实验修改后运行结果截图

当输入4 5时，测试结果如图4-2所示

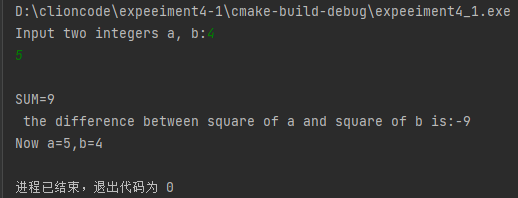


图4-2 实验修改后运行结果截图

当输入-3 4时，测试结果如图4-3所示

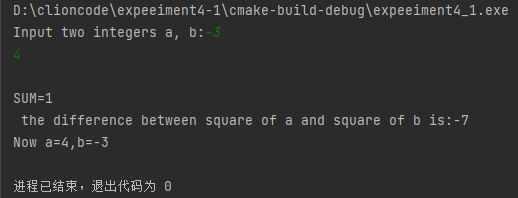


图4-3 实验修改后运行结果截图

**2．程序修改替换**

下面是用函数实现求三个数中最大数、计算两浮点数之和的程序。在这个源程序中存在若干语法和逻辑错误。

要求：（1）对这个例子程序进行调试修改，使之能够正确完成指定任务；

（2）用带参数的宏替换函数max，来实现求最大数的功能。

/\*实验4-2程序修改替换题程序\*/

#include<stdio.h>

int main(void)

{

int a, b, c;

float d, e;

printf("Input three integers:");

scanf("%d %d %d",&a,&b,&c);

printf("\nThe maximum of them is %d\n",max(a,b,c));

printf("Input two floating point numbers:");

scanf("%f %f",&d,&e);

printf("\nThe sum of them is %f\n",sum(d,e));

return 0;

}

int max(int x, int y, int z)

{

int m=z;

if (x>y)

if(x>z) m=x;

else

if(y>z) m=y;

return m;

}

float sum(float x, float y)

{

return x+y;

}

**解答：**

1. 1）错误修改：

1.将函数的定义提前到main函数之前。

2.max函数求最大值有误，应改为

int max(int x, int y, int z)

{

int m=z;

if (x>y) {

if (x > z) m = x;

else m = z;

}

else{

if(y<z) m=z;

else m=y;

}

return m;

}

2）修改后的程序如下：

#include<stdio.h>

float sum(float x, float y)

{

return x+y;

}

int max(int x, int y, int z)

{

int m=z;

if (x>y) {

if (x > z) m = x;

else m = z;

}

else{

if(y<z) m=z;

else m=y;

}

return m;

}

int main(void)

{

int a, b, c;

float d, e;

printf("Input three integers:");

scanf("%d %d %d",&a,&b,&c);

printf("\nThe maximum of them is %d\n",max(a,b,c));

printf("Input two floating point numbers:");

scanf("%f %f",&d,&e);

printf("\nThe sum of them is %f\n",sum(d,e));

return 0;

}

3）测试：

当输入4,5,6三个数以及3.2,4.8两个浮点数时，测试结果如图4-4所示

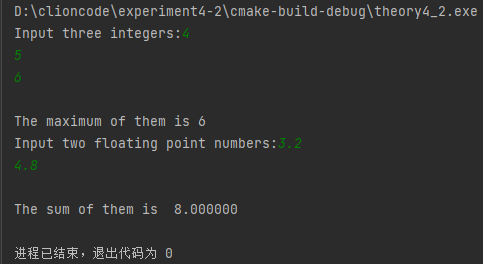


图4-4 程序修改题1运行结果截图

当输入0,0,2三个数以及2.33,4.66两个浮点数时，测试结果如图4-5所示

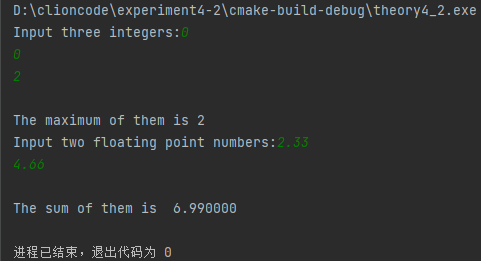


图4-5 程序修改题1运行结果截图

1. 修改：
2. 思路：利用条件运算求出最大值
3. 修改后的程序如下：

#include<stdio.h>

#define MAX(a,b,c) (((a)>(b))?((a)>(c)?a:c):((b)<(c)?c:b))

float sum(float x, float y)

{

return x+y;

}

int main(void)

{

int a, b, c;

float d, e;

printf("Input three integers:");

scanf("%d %d %d",&a,&b,&c);

printf("\nThe maximum of them is %d\n",MAX(a,b,c));

printf("Input two floating point numbers:");

scanf("%f %f",&d,&e);

printf("\nThe sum of them is %f\n",sum(d,e));

return 0;

}

1. 测试：
2. 测试数据如表4-1所示

表4-1 程序修改题2的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 |
| a,b,c,d,e |
| 用例1 | 12 14 3 3.98 4.22 | The maximum of them is 14  The sum of them is 7.980000 |
| 用例2 | -3 -5 -7 3.22 5.345 | The maximum of them is -3  The sum of them is 8.565000 |

（b）对应测试用例1的运行结果如图4-6所示

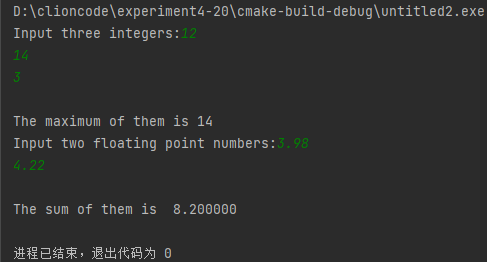


图4-6 程序修改题2测试用例1运行结果截图

对应测试用例2的运行结果如图4-7所示

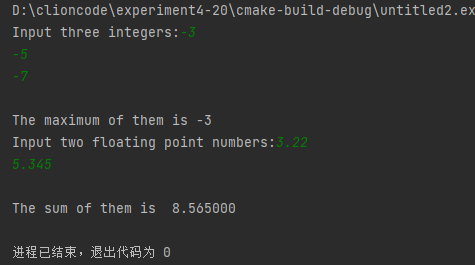


图4-7 程序修改题2测试用例2运行结果截图

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

**3．跟踪调试**

下面程序利用R计算圆的面积s，以及面积s的整数部分。现要求：

（1）修改程序，使程序编译通过且能运行；

（2）单步执行。进入函数integerl\_fraction时，watch窗口中x为何值？在返回main时, watch窗口中i为何值？

（3）修改程序，使程序能输出面积s值的整数部分（要求四舍五入），不会输出错误信息assertion failed。

/\*实验4-3跟踪调试题程序利用R计算圆的面积s\*/

#define R

int main(void)

{

float r, s;

int s\_integer=0;

printf ("Input a number: ");

scanf("%f",&r);

#ifdef R

s=3.14159\*r\*r;

printf("Area of round is: %f\n",s);

s\_integer=integer\_fraction(s);

assert((s-s\_integer)<0.5);

printf("The integer fraction of area is %d\n", s\_integer);

#endif

return 0;

}

int integer\_fraction(float x)

{

int i=x;

return i;

}

**解答：**

1)错误修改

1.缺少#include <stdio.h>和#include<assert.h>。

2.取整函数的定义应在main函数之前。

2)修改后的程序如下：

#define R

#include <stdio.h>

int integer\_fraction(float x)

{

int i=x;

return i;

}

int main(void)

{

float r, s;

int s\_integer=0;

printf ("Input a number: ");

scanf("%f",&r);

#ifdef R

s=3.14159\*r\*r;

printf("Area of round is: %f\n",s);

s\_integer=integer\_fraction(s);

assert((s-s\_integer)<0.5);

printf("The integer fraction of area is %d\n", s\_integer);

#endif

return 0;

}

3)测试：如图4-8所示

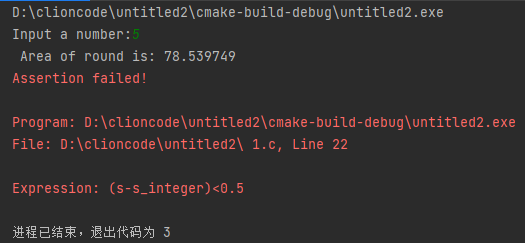


图4-8 实验4-3修改后运行结果截图

发现出现错误信息assertion failed。

1. 观察：

当输入5时

进入函数integerl\_fraction时，watch窗口中x为78.5397491。

在返回main时, watch窗口中i为78。

1. 程序修改
2. 程序清单如下：

#define R

#include <stdio.h>

float integer\_fraction(float x)

{

int i=x;

return i;

}

int main(void)

{

float r, s;

int s\_integer=0;

printf ("Input a number: ");

scanf("%f",&r);

#ifdef R

s=3.14159\*r\*r;

printf("Area of round is: %f\n",s);

s\_integer=integer\_fraction(s);

if((s-s\_integer)<0.5)

printf("The integer fraction of area is %d\n", s\_integer);

else

printf("The integer fraction of area is %d\n", s\_integer+1);

#endif

return 0;

}

1. 测试：

（a）测试数据如表4-2所示

表4-2 实验4-3修改后的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例1 | 5 | Area of round is: 78.539749  The integer fraction of area is 79 | Area of round is: 78.539749  The integer fraction of area is 79 |
| 用例2 | 7 | Area of round is: 153.937912  The integer fraction of area is 154 | Area of round is: 153.937912  The integer fraction of area is 154 |
| 用例3 | 11 | Area of round is: 380.132385  The integer fraction of area is 380 | Area of round is: 380.132385  The integer fraction of area is 380 |

（b）对应测试用例运行结果如下图所示

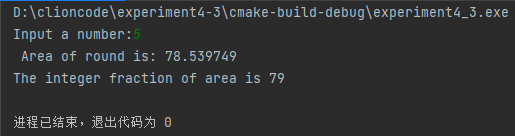


图4-9 实验4-3修改后用例1的运行结果

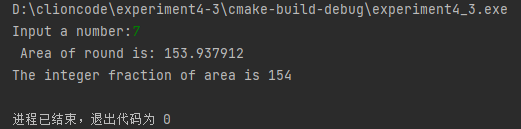


图4-10 实验4-3修改后用例2的运行结果

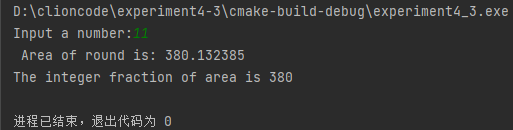


图4-11 实验4-3修改后用例3的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

**4．程序设计**

（1）三角形的面积是，其中，a,b,c为三角形的三边，要求编写程序用带参数的宏来计算三角形的面积。定义两个带参数的宏，一个用来求s，另一个用来求area。

**解答：**

（1）解题思路：如图4-12所示



图4-12 实验设计题1的程序设计流程图

（2）程序清单：

#include <stdio.h>

#include <math.h>

#define s(a,b,c) ((a+b+c)/2)

#define area(a,b,c,s) (sqrt((s)\*(s-a)\*(s-b)\*(s-c)))

int main() {

float a,b,c,s;

printf("Input a b c plz:\n");

scanf("%f%f%f",&a,&b,&c);

s=s(a,b,c);

printf("area = %f",area(a,b,c,s));

return 0;

}

1. 测试：
2. 测试数据如表4-3所示

表4-3 程序设计题1的测试数636E

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| a，b，c |
| 用例1 | 3,4,5 | 6.000000 | 6.000000 |
| 用例2 | 3,3,3 | 3.897114 | 3.897114 |
| 用例3 | 2.5,3.5,4.5 | 4.353070 | 4.353070 |

（b）对应测试用例运行结果如下图所示

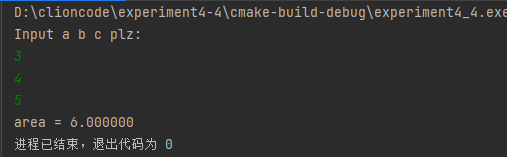


图4-13 程序设计题1用例1的运行结果

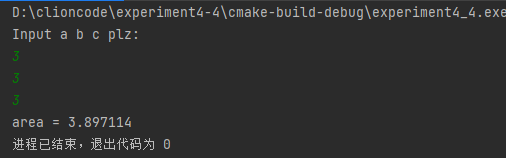


图4-14 程序设计题1用例2的运行结果

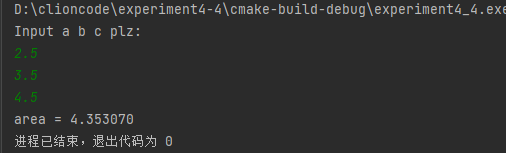


图4-15 程序设计题1用例3的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

（2）用条件编译方法来编写程序。输入一行英文字符序列，可以任选两种方式之一输出：一为原文输出；二为变换字母的大小写后输出。例如小写‘a’变成大写‘A’，大写‘D’变成小写‘d’，其他字符不变。用#define命令控制是否变换字母的大小写。例如，#define CHANGE 1 则输出变换后的文字，若#define CHANGE 0则原文输出。

**解答：**

1. 解题思路：如图4-16所示。



图4-16 程序设计题2的程序设计流程图

1. 程序清单如下：

#include <stdio.h>

#define CHANGE 0

#if CHANGE

int main(){

printf("Input text plz:\n");

char c;

while((c=getchar())!='\n')

if((c<='Z')&&(c>='A')) putchar(c-'A'+'a');

else putchar(c-'a'+'A');

return 0;

}

#else

int main() {

printf("Input text plz:\n");

char c;

while((c=getchar())!='\n')

putchar(c);

return 0;

}

#endif

1. 测试：
2. 测试数据如表4-4所示。

表4-4 程序设计题2的测试数据

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测 试用 例 | 程序输入 | | 理论结果 | 运行结果 |
| CHANGE | c |
| 用例1 | 0 | Sjfslj | Sjfslj | Sjfslj |
| 用例2 | 1 | Sjfslj | sJFSLJ | sJFSLJ |
| 用例3 | 0 | SSSf ddd | SSSf ddd | SSSf ddd |
| 用例4 | 1 | SSSf ddd | sssF DDD | sssF DDD |

1. 对应测试案例运行结果如下图所示

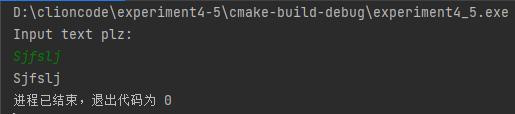


图4-17 程序设计题2用例1的运行结果

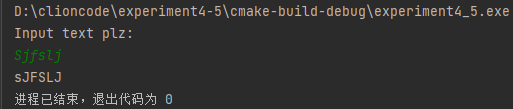


图4-18 程序设计题2用例2的运行结果

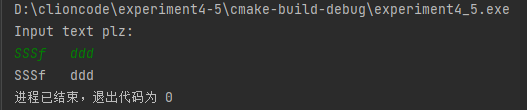


图4-19 程序设计题2用例3的运行结果

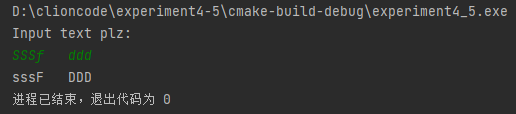


图4-20 程序设计题2用例4的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

（3）假设一个C程序由file1.c和file2.c两个源文件及一个file.h头文件组成，file1.c、file2.c和file.h的内容分别如下所述。试编辑该多文件C程序，补充file.h头文件内容，然后编译和链接。然后运行最后生成的可执行文件。

/\*源文件file1.c的内容\*/

#include "file.h"

int x,y; /\* 外部变量的定义性说明 \*/

char ch; /\* 外部变量的定义性说明 \*/

int main(void)

{

x=10;

y=20;

ch=getchar();

printf("in file1 x=%d,y=%d,ch is %c\n",x,y,ch);

func1();

return 0;

}

/\*源文件file2.c的内容为：\*/

#include "file.h"

void func1(void)

{

x++;

y++;

ch++;

printf("in file2 x=%d,y=%d,ch is %c\n",x,y,ch);

}

**解答：**

1. 解题思路：

由于getchar，printf等函数的定义均包含在<stdio.h>头文件中

则在file.c中应包含<stdio.h>头文件，并声明x，y，ch，func1等类型。

1. 头文件程序清单：

#ifndef EXPERIMENT4\_6\_FILE\_H

#define EXPERIMENT4\_6\_FILE\_H

#include <stdio.h>

extern void func1(void);

extern int x,y;

extern char ch;

#endif

1. 测试：
2. 测试数据如表4-5所示。

表4-5 程序设计题3的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 |
| 用例1 | a | in file1 x=10,y=20,ch is a  in file2 x=11,y=21,ch is b |
| 用例2 | Z | in file1 x=10,y=20,ch is Z  in file2 x=11,y=21,ch is [ |
| 用例3 | 2 | in file1 x=10,y=20,ch is 2  in file2 x=11,y=21,ch is 3 |
| 用例4 | ‘，’ | in file1 x=10,y=20,ch is ,  in file2 x=11,y=21,ch is - |

1. 对应测试用例运行结果如下图所示

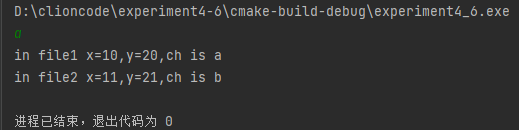


图4-21 程序设计题3用例1的运行结果

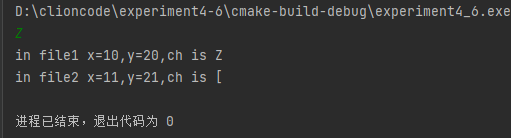


图4-22 程序设计题3用例2的运行结果

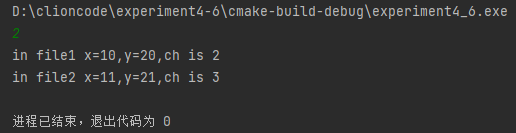


图4-23 程序设计题3用例3的运行结果

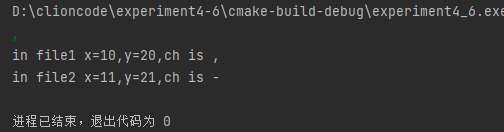


图4-24 程序设计题3用例4的运行结果

## 4.3 实验小结

本次实验过程中，较前两次来说较顺利，不知是题目变简单了，还是能力提升了。虽然较顺利，但还是发现了不少问题，如（）没有打到位，自己还没有发现，导致卡壳了很长一段时间。其中还是有几个不懂的问题，比如assert断言的修改，还是没有完全搞懂。

还有对多文件的程序编写，extern的运用都不太熟练，今后要更加加强这方面的练习。

# 实验5 数组实验

## 5.1实验目的

（1）掌握数组的说明、初始化和使用。

（2）掌握一维数组作为函数参数时实参和形参的用法。

（3）掌握字符串处理函数的设计，包括串操作函数及数字串与数之间转换函数实现算法。

（4）掌握基于分治策略的二分查找算法和选择法排序算法的思想，以及相关算法的实现。

## 5.2 实验内容及要求

* + 1. **源程序改错与跟踪调试**

在下面所给的源程序中，函数strcate(t,s)的功能是将字符串s连接到字符串t的尾部；函数strdelc(s,c)的功能是从字符串s中删除所有与给定字符c相同的字符，程序应该能够输出如下结果：

Programming Language

ProgrammingLanguage Language

ProgrmingLnguge

跟踪和分析源程序中存在的问题，排除程序中的各种逻辑错误，使之能够输出正确的结果。

1. 单步执行源程序。跟踪进入strcate时，观察字符数组t和s中的内容，分析结果是否正确。当单步执行光条刚落在第二个while语句所在行时，i为何值？t[i]为何值？分析该结果是否存在问题。当单步执行光条落在strcate函数块结束标记即右花括 号“}”所在行时，字符数组t和s分别为何值？分析是否实现了字符串连接。
2. 跟踪进入函数strdelc时，观察字符数组s中的内容和字符c的值，分析结果是否正确。单步执行for语句过程中，观察字符数组s, j和k值的变化，分析该结果是否存在问题。当单步执行光条落在strdelc函数块结束标记“}”所在行时，字符串s为何值？分析是否实现了所要求的删除操作。

/\*实验5-1程序改错与跟踪调试题程序\*/

1. #include<stdio.h>
2. void strcate(char [],char []);
3. void strdelc(char [],char );
4. int main(void)
5. {
6. char a[]="Language", b[]="Programming";
7. printf("%s %s\n", b,a);
8. strcate(b,a);
9. printf("%s %s\n",b,a);
10. strdelc(b, 'a');
11. printf("%s\n",b);
12. return 0;
13. }
14. void strcate(char t[],char s[])
15. {
16. int i = 0, j = 0;
17. while(t[i++]) ;
18. while((t[i++] = s[j++] )!= '\0');
19. }
20. void strdelc(char s[], char c)
21. {
22. int j,k;
23. for(j=k=0; s[j] != '\0'; j++)
24. if(s[j] != c) s[k++] = s[j];
25. }

**解答：**

1. 跟踪进入strcate时，字符数组t和s中的内容分别为"Programming"和"Language"。当单步执行光条刚落在第二个while语句所在行时，i为12（如图5-1所示），t[i]为'L'。当单步执行光条落在strcate函数块结束标记即右花括号“}”所在行时，字符数组t和s分别为"Programming"和"Language"（如图5-2所示），未实现字符串连接。

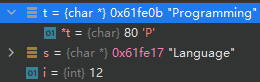


图5-1 实验5-1光条刚落在第二个while语句所在行时i的值截图



图5-2 实验5-1光条落在strcate函数块结束标记所在行时字符数组t和s内容截图

* 1. 跟踪进入函数strdelc时，字符数组s中的内容为"Programming"，字符c的值为'a'。单步执行for语句过程中，字符数组s, j和k值的变化如图5-3所示。当单步执行光条落在strdelc函数块结束标记“}”所在行时，字符s为"Progrmmingg"，（如图5-4所示）未实现删除功能。

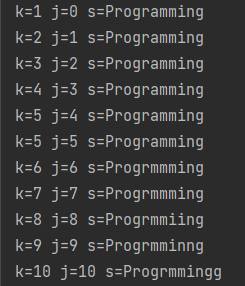


图5-3 实验5-1函数strdelc中单步执行for语句过程中，字符数组s, j和k值截图



图5-4 实验5-1函数strdelc结束时字符串s截图

1. 错误修改：
   1. 第6行，b数组更改范围，改为20。
   2. 第18行，添加i--。
   3. 第24行后添加s[k]='\0';。

修改后，源程序的清单如下：

#include<stdio.h>

void strcate(char [],char []);

void strdelc(char [],char );

int main(void)

{

char a[]="Language", b[20]="Programming";

printf("%s %s\n", b,a);

strcate(b,a);

printf("%s %s\n",b,a);

strdelc(b, 'a');

printf("%s\n",b);

return 0;

}

void strcate(char t[],char s[])

{

int i = 0, j = 0;

while(t[i++] );

i--;

while((t[i++] = s[j++] )!= '\0');

}

void strdelc(char s[], char c)

{

int j,k;

for(j=k=0; s[j] != '\0'; j++)

if(s[j] != c) s[k++] = s[j];

s[k]='\0';

}

1. 错误修改后运行结果如图5-4所示。

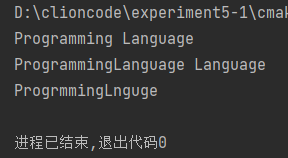


图5-5 实验5-1修改后运行结果截图

* + 1. **源程序完善和修改替换**

1. 下面的源程序用于求解瑟夫问题：M个人围成一圈，从第一个人开始依次从1至N循环报数，每当报数为N时报数人出圈，直到圈中只剩下一个人为止。

①请在源程序中的下划线处填写合适的代码来完善该程序。

#include<stdio.h>

#define M 10

#define N 3

int main(void)

{

int a[M], b[M]; /\* 数组a存放圈中人的编号，数组b存放出圈人的编号 \*/

int i, j, k;

for(i = 0; i < M; i++) /\* 对圈中人按顺序编号1—M \*/

a[i] = i + 1;

for(i = M, j = 0; i > 1; i--){

/\* i表示圈中人个数，初始为M个，剩1个人时结束循环；j表示当前报数人的位置 \*/

for(k = 1; k <= N; k++) /\* 1至N报数 \*/

if(++j > i - 1) j = 0;/\* 最后一个人报数后第一个人接着报，形成一个圈 \*/

b[M-i] = j ? \_\_\_\_\_\_\_:\_\_\_\_\_\_; /\* 将报数为N的人的编号存入数组b \*/

if(j)

for(k = --j; k < i; k++) /\* 压缩数组a，使报数为N的人出圈 \*/

\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

}

for(i = 0;i < M-1; i++) /\* 按次序输出出圈人的编号 \*/

printf("%6d", b[i]);

printf("%6d\n", a[0]); /\* 输出圈中最后一个人的编号 \*/

return 0;

}

1. 上面的程序中使用数组元素的值表示圈中人的编号，故每当有人出圈时都要压缩数组，这种算法不够精炼。如果采用做标记的办法，即每当有人出圈时对相应数组元素做标记，从而可省掉压缩数组的时间，这样处理效率会更高一些。请采用做标记的办法修改程序，并使修改后的程序与原程序具有相同的功能。

**解答：**

1. 三处填空分别为：
2. a[j-1]
3. a[i-1]
4. a[k]=a[k+1]
5. （1）实验思路如下：
6. 先定义a，b两个一维数组，利用for循环给a中每个元素标号。
7. 再用for循环进行报号。
8. 利用for循环，每3个报一次，并进行判断，若报号者为第一个人，则将位置j记为0，若报号者已经报过号了，则此次不计入报号，并从下一个开始继续报号。
9. 将每次报号的人存入b数组，并将a数组中，报号者记为0。
10. 利用for循环将b中的报号者序号依次输出，最后输出a中唯一不为0的序号，即为获胜者。
11. 程序清单如下：

#include<stdio.h>

#define M 10

#define N 3

int main(void)

{

int a[M], b[M],c[M-1];

int i, j, k,t;

for(i = 0; i < M; i++)

a[i] = i + 1;

for(i = M, j = 0; i > 1; i--) {

for (k = 1; k <= N; k++)

{

if(a[j]==0) k--;

if (++j > M - 1) j = 0;

}

b[M - i] = j ? a[j - 1] : a[M - 1];

if(j)

a[j-1]=0;

else

a[M-1]=0;

}

for(i = 0;i < M-1; i++)

printf("%6d", b[i]);

for(i=0;i<M-1;i++)

if(a[i]!=0) printf("%6d",a[i]);

return 0;

}

1. 测试。
2. 测试数据如表5-1所示。

表5-1 实验5-2的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 理论结果 | 运行结果 |
|
| 用例1 | 3 6 9 2 7 1 8 5 10 4 | 3 6 9 2 7 1 8 5 10 4 |

1. 对应测试用例1的运行结果如图5-6所示。

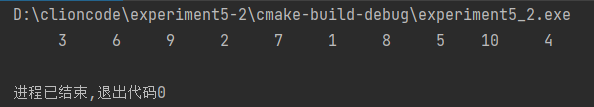


图5-6 实验5-2用例1的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

* + 1. **程序设计**

1. 输入一个整数，将它在内存中二进制表示的每一位转化成对应的数字字符并且存放到一个字符数组中，然后输出该整数的二进制表示。

**解答：**

1. 实验思路如下：
2. 定义一个大小为32的字符数组，并初始化为‘0’。
3. 定义常量mask 0x00000001。
4. 使用for循环，每次循环取对应数字的一位并存储在字符数组中，并将该数字右移一位。再将该数组顺序打印，即为所得。
5. 程序清单如下：

#include <stdio.h>

int main() {

int m;

char a[33]={'0'};

scanf("%d",&m);

const int mask=0x00000001;

int i,j;

for(i=31;i>=0;i--)

{

a[i]=(m&mask)+'0';

m>>=1;

}

for(j=0;j<32;j++)

printf("%c",a[j]);

return 0;

}

1. 测试。
2. 测试数据如表5-2所示。

表5-2 程序设计题1的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测 试用 例 | 程 序 输 入 | 理论结果 | 实际结果 |
| x |
| 用例1 | 10 | 00000000000000000000000000001010 | 00000000000000000000000000001010 |
| 用例2 | -33 | 11111111111111111111111111011111 | 11111111111111111111111111011111 |
| 用例3 | 0 | 00000000000000000000000000000000 | 00000000000000000000000000000000 |

1. 对应测试用例1的运行结果如图5-7所示。

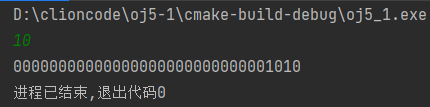


图5-7 程序设计题1用例1的运行结果截图

对应测试用例2的运行结果如图5-8所示。

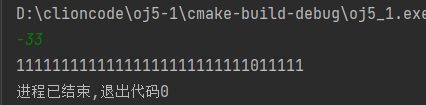


图5-8 程序设计题1用例2的运行结果截图

对应测试用例3的运行结果如图5-9所示。

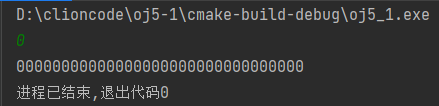


图5-9 程序设计题1用例3的运行结果截图

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

1. 编写一个C程序，要求采用模块化程序设计思想，将相关功能用函数实现，并提供菜单选项。该程序具有以下功能：  
   ①“成绩输入”，输入n个学生的姓名和C语言课程的成绩。  
   ②“成绩排序”，将成绩按从高到低的次序排序，姓名同时进行相应调整。成绩相同的，按照输入先后次序排列。  
   ③“成绩输出”，输出排序后所有学生的姓名和C语言课程的成绩。  
   ④“成绩查找”，输入一个C语言课程成绩值，用二分查找进行搜索。如果查找到有该成绩，则输出该成绩学生的姓名和C语言课程的成绩；否则，输出提示“not found!”。

**解答：**

1. 实验思路如下：
2. 定义变量i，用于接收输入的菜单值，定义变量n，用于表示需存储的数量，定义字符数组和整型数组用来存储姓名和成绩。
3. 定义INPUT函数，并将n，整形数组和字符数组作为参数，利用for循环将姓名与成绩存储进数组中，存储完毕，输出“n records were input!”。
4. 定义SORT函数，并将n，整形数组和字符数组作为参数，利用冒泡排序，将整型数组又小到大排序，并同时用strcpy函数更改姓名数组，完成后输出“Reorder finished!”
5. 定义OUTPUT函数，并将n，整形数组和字符数组作为参数，利用for循环，输出排序后的姓名与成绩。
6. 定义FIND函数，利用二分查找，查找输入的成绩，若找到则输出成绩与对应的姓名，若没找到，则输出“not found!”。
7. 主函数利用switch将4个函数分割，并用while函数实现不断的输入。
8. 程序清单如下：

#include <stdio.h>

#include "string.h"

void INPUT(int n,char b[][100],int a[])

{

int i;

for(i=0;i<n;i++)

scanf("%s%d",b[i],&a[i]);

printf("%d records were input!\n",i);

}

void SORT(int n,char b[][100],int a[])

{

int i,t;

char c[100];

for(i=0;i<n-1;i++)

{

for(int j=0;j<n-1;j++)

{

if (a[j]>=a[j + 1])

{

t = a[j], a[j] = a[j + 1], a[j + 1] = t;

strcpy(c, b[j]);

strcpy(b[j], b[j + 1]);

strcpy(b[j + 1], c);

}

}

}

printf("Reorder finished!\n");

}

void OUTPUT(int n,char b[][100],int a[])

{

for(int i=n-1;i>=0;i--)

{

printf("%s %d\n",b[i],a[i]);

}

}

void FIND(int x,int n,char b[][100],int a[])

{

int front=0, back=n-1, middle;

while(front<=back){

middle=(front+back)/2;

if(x<a[middle]) back=middle-1;

else if(x>a[middle]) front=middle+1;

else{

printf("%s %d\n", b[middle],a[middle]);

break;

}

}

if(x != a[middle]) printf("not found!\n");

}

int main(){

int i,n,k;

while ((scanf("%d",&i))!=0)

{

if(i==1)

{

scanf("%d",&n);

}

static int t;

t=n;

int a[20];

char b[20][100];

switch (i)

{

case 0:

return 0;

case 1:

INPUT(t,b,a);break;

case 2:

SORT(t,b,a);break;

case 3:

OUTPUT(t,b,a);break;

case 4:

scanf("%d",&k);

FIND(k,t,b,a);break;

}

}

return 0;

}

1. 测试。
   1. 测试数据如表5-3所示。

表5-3 程序设计题2的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 |
| 用例1 | 输入0. | 结束程序。 |
| 用例2 | 输入1，输入5；  再输入  a 80  b 77  c 88  d 98  e 78；  输入2；  输入3；  输入4，输入78；  输入4，输入100；  输入0。 | 依次输出  5 records were input!；  Reorder finished!；  d 98  c 88  a 80  e 78  b 77；  e 78；  not found!；  结束程序。 |

* 1. 对应测试用例1的运行结果如图5-10所示。

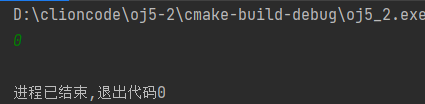


图5-10 程序设计题2用例1的运行结果截图

对应测试用例2的运行结果如下图所示。

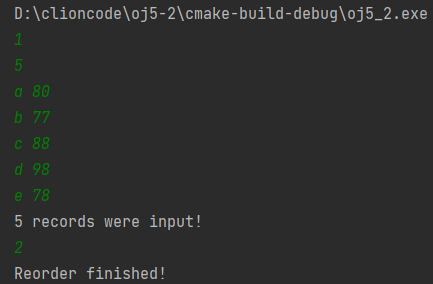


图5-11 程序设计题2用例2的运行结果截图

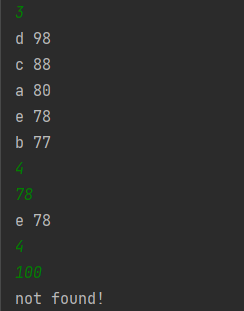


图5-12 程序设计题2用例2的运行结果截图

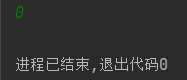


图5-13 程序设计题2用例2的运行结果截图

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

1. 求解N皇后问题，即在N×N的棋盘上摆放N个皇后，要求任意两个皇后不能在同一行、同一列、同一对角线上。输入棋盘的大小N（N取值1-10），如果能满足摆放要求，则输出所有可能的摆放法的数量，否则输出“无解”。

**解答：**

1. 实验思路如下：
2. 定义全局变量count=0，size，和二维数组a[10][10]作为棋盘。
3. 定义check函数用来检测该位置的左斜上和右斜上和上方是否有皇后存在
4. 定义ok=1。
5. 若该位置不能放置皇后，则ok=0。
6. 返回ok的值。
7. 定义solve函数，将行数row作为参数，从第一行第一列进行check，若check为1，则执行solve（row+1），并将该位置标记为1，solve（row+1）结束后，将该位置还原为0，实现回溯，以size>=row为递归结束条件，每次递归结束都令count++。
8. 主函数中，接收输入的size，令solve从第0行开始执行，最后输出count的值。
9. 程序清单如下：

#include <stdio.h>

int size,count=0;

int a[10][10]={0};

int check(int row,int col);

void solve(int row){

int i;

if(row>=size) count++;

else

for(i=0;i<size;i++)

{

if(check(row,i))

{

a[row][i]=1;

solve(row+1);

a[row][i]=0;

}

}

}

int check(int row,int col)

{

int ok=1;

for(int i=0;i<row&&ok;i++)

{

if(a[i][col]==1)

ok=0;

}

for(int j=1;j<=col&&ok&&j<=row;j++)

{

if(a[row-j][col-j]==1)

ok=0;

}

for(int m=1;row-m>=0&&ok&&col+m<size;m++)

{

if(a[row-m][col+m]==1)

ok=0;

}

return ok;

}

int main(){

scanf("%d",&size);

solve(0);

if(count!=0) printf("%d",count);

else printf("无解");

return 0;

}

1. 测试。
   1. 测试数据如表5-4所示。

表5-4 程序设计题3的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例1 | 1 | 1 | 1 |
| 用例2 | 2 | 无解！ | 无解！ |
| 用例3 | 4 | 2 | 2 |
| 用例4 | 5 | 10 | 10 |
| 用例5 | 7 | 40 | 40 |
| 用例6 | 10 | 724 | 724 |

* 1. 对应测试用例1的运行结果如图5-14所示。

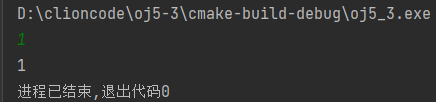


图5-14 程序设计题3用例1的运行结果截图

对应测试用例2的运行结果如图5-15所示。

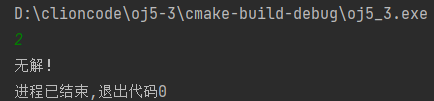


图5-15 程序设计题3用例2的运行结果截图

对应测试用例3的运行结果如图5-16所示。

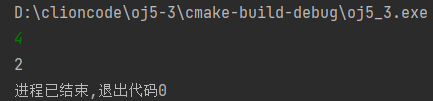


图5-16 程序设计题3用例3的运行结果截图

对应测试用例4的运行结果如图5-17所示。

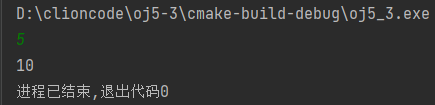


图5-17 程序设计题3用例4的运行结果截图

对应测试用例5的运行结果如图5-18所示。

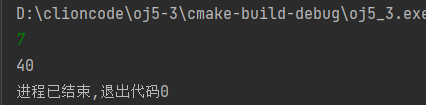


图5-18 程序设计题3用例5的运行结果截图

对应测试用例6的运行结果如图5-19所示。

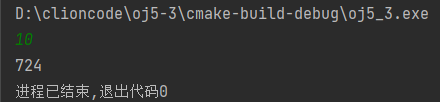


图5-19 程序设计题3用例6的运行结果截图

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

## 5.3 实验小结

本次实验，首先对字符串处理函数的原理和应用加深了印象，包括数组的范围等问题，都有较深体会。其次在头歌的第二题中，应该算是我第一次接触类似完整程序的代码，其中对各个函数间的关系让我有很深刻的认识，也让我知道一些微小的地方，也可能对整个程序有着很大的影响，比如>和>=的区别。最后就是N皇后的问题，回溯算法是我仍然需要好好研究的一个问题。

实验6 指针实验

## 6.1实验目的

（1）熟练掌握指针的说明、赋值、使用。

（2）掌握用指针引用数组的元素，熟悉指向数组的指针的使用。

（3）熟练掌握字符数组与字符串的使用，掌握指针数组及字符指针数组的用法。

（4）掌握指针函数与函数指针的用法。

（5）掌握带有参数的main函数的用法。

## 6.2实验题目及要求

**1、源程序改错题**

在下面所给的源程序中，函数strcopy(t, s)的功能是将字符串s复制给字符串t，并且返回串t的首地址。请单步跟踪程序，根据程序运行时出现的现象或观察到的字符串的值，分析并排除源程序的逻辑错误，使之能按照要求输出如下结果：

Input a string:

programming↙ （键盘输入）

programming

Input a string again:

language↙ （键盘输入）

language

1. #include<stdio.h>
2. char \*strcopy(char \*, const char \*);
3. int main(void)
4. {
5. char \*s1, \*s2, \*s3;
6. printf("Input a string:\n", s2);
7. scanf("%s", s2);
8. strcopy(s1, s2);
9. printf("%s\n", s1);
10. printf("Input a string again:\n", s2);
11. scanf("%s", s2);
12. s3 = strcopy(s1, s2);
13. printf("%s\n", s3);
14. return 0;
15. }

/\*将字符串s复制给字符串t，并且返回串t的首地址\*/

1. char \* strcopy(char \*t, const char \*s)
2. {
3. while(\*t++ = \*s++);
4. return (t);
5. }

**解答：**

1. 错误修改
2. 第5行应用数组存储输入字符串，并将指针指向数组，否则指针 为空，应该为：char a[20],b[20],c[20];char \*s1=a,\*s2=b,\*s3=c;
3. 第16行strcopy函数有误，应改为：

char \* strcopy(char \*t, char \*s)

{

int i;

for(i=0;\*(s+i)!='\0';i++)

\*(t+i)=\*(s+i);

\*(t+i)='\0';

return (t);

1. 修改后，源程序清单如下：

#include<stdio.h>

char \*strcopy(char \*, char \*);

int main(void)

{

char \*s1, \*s2, \*s3;

char a[20],b[20],c[20];

s1=a,s2=b,s3=c;

printf("Input a string:\n");

scanf("%s", b);

strcopy(s1, s2);

printf("%s\n", s1);

printf("Input a string again:\n");

scanf("%s", b);

s1=a,s2=b;

s3 = strcopy(s1, s2);

printf("%s\n", s3);

return 0;

}

/\*将字符串s复制给字符串t，并且返回串t的首地址\*/

char \* strcopy(char \*t, char \*s)

{

int i;

for(i=0;\*(s+i)!='\0';i++)

\*(t+i)=\*(s+i);

\*(t+i)='\0';

return (t);

}

1. 修改后运行结果如图6-1所示。

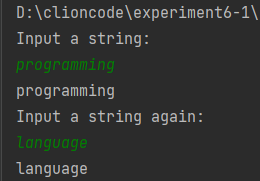


图6-1 程序改错题运行结果截图

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

**2、源程序完善、修改替换题**

（1）下面程序中函数strsort用于对字符串进行升序排序，在主函数中输入N个字符串（字符串长度不超过49）存入通过malloc动态分配的存储空间，然后调用strsort对这N个串按字典序升序排序。

①请在源程序中的下划线处填写合适的代码来完善该程序。

#include<stdio.h>

#include< \_\_\_\_\_\_\_\_\_>

#include<string.h>

#define N 4

/\*对指针数组s指向的size个字符串进行升序排序\*/

void strsort(char \*s[], int size)

{

char\_\_\_\_\_;

int i, j;

for(i=0; i<size-1; i++)

for (j=0; j<size-i-1; j++)

if (\_\_\_\_\_\_\_\_\_\_\_)

{

temp = s[j];

\_\_\_\_\_\_\_\_\_\_\_;

s[j+1] = temp;

}

}

int main()

{

int i;

char \*s[N], t[50];

for (i=0; i<N; i++)

{

gets(t);

s[i] = (char \*)malloc(strlen(t)+1);

strcpy(\_\_\_\_\_\_\_);

}

strsort(\_\_\_\_\_\_\_\_);

for (i=0; i<N; i++) {puts(s[i]); free(s[i]);}

return 0;

}

1. **解答：**
2. 填空：
3. stdlib.h
4. \*temp
5. s[j]>s[j+1]
6. s[j]=s[j+1]
7. s[i],t
8. s,N
9. 补全后源程序清单如下：

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define N 4

/\*对指针数组s指向的size个字符串进行升序排序\*/

void strsort(char \*s[], int size)

{

char\* temp;

int i, j;

for(i=0; i<size-1; i++)

for (j=0; j<size-i-1; j++)

if (\*s[j]>\*s[j+1])

{

temp = s[j];

s[j]=s[j+1];

s[j+1] = temp;

}

}

int main()

{

int i;

char \*s[N], t[50];

for (i=0; i<N; i++)

{

gets(t);

s[i] = (char \*)malloc(strlen(t)+1);

strcpy(s[i],t);

}

strsort(s,N);

for (i=0; i<N; i++) {puts(s[i]); free(s[i]);}

return 0;

}

1. 测试：
2. 测试数据如表6-1所示：

表6-1 源程序完善与修改替换题（1）测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例1 | aba  cdc  r09  j4e | aba  cdc  j4e  r09 | aba  cdc  j4e  r09 |
| 用例2 | lhy  fzx  love  forgood | fzx  forgood  lhy  love | fzx  forgood  lhy  love |
| 用例3 | qq  wechat  ins  telegram | ins  qq  telegram  wechat | ins  qq  telegram  wechat |

1. 运行结果如下图所示：

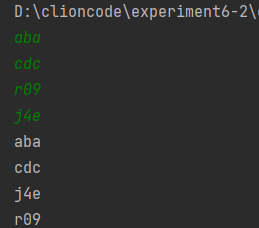


图6-2 源程序完善与修改替换题1测试用例1运行结果截图

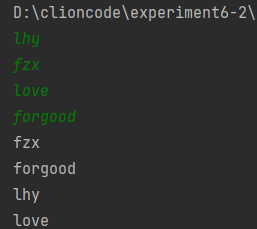


图6-3 源程序完善与修改替换题1测试用例2运行结果截图

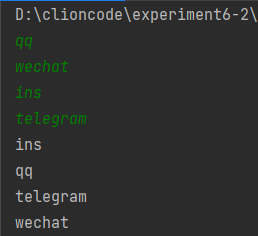


图6-4 源程序完善与修改替换题1测试用例3运行结果截图

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

②数组作为函数参数其本质类型是指针。例如，对于形参char \*s[]，编译器将其解释为char \*\*s，两种写法完全等价。请用二级指针形参重写strsort函数，并且在该函数体的任何位置都不允许使用下标引用。

1. 解答：
2. 源程序清单如下：

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define N 4

/\*对指针数组s指向的size个字符串进行升序排序\*/

void strsort(char \*\*s, int size)

{

char\* temp;

int i, j;

for(i=0; i<size-1; i++)

for (j=0; j<size-i-1; j++)

if (\*\*(s+j)>\*\*(s+j+1))

{

temp = \*(s+j);

\*(s+j)=\*(s+j+1);

\*(s+j+1) = temp;

}

}

int main()

{

int i;

char \*s[N], t[50];

for (i=0; i<N; i++)

{

gets(t);

s[i] = (char \*)malloc(strlen(t)+1);

strcpy(s[i],t);

}

strsort(s,N);

for (i=0; i<N; i++) {puts(s[i]); free(s[i]);}

return 0;

}

1. 测试：

a） 测试数据如表6-2所示：

表6-2 源程序完善与修改替换题（1）测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例1 | aba  cdc  r09  j4e | aba  cdc  j4e  r09 | aba  cdc  j4e  r09 |
| 用例2 | lhy  fzx  love  forgood | fzx  forgood  lhy  love | fzx  forgood  lhy  love |
| 用例3 | qq  wechat  ins  telegram | ins  qq  telegram  wechat | ins  qq  telegram  wechat |

1. 运行结果如下图所示：

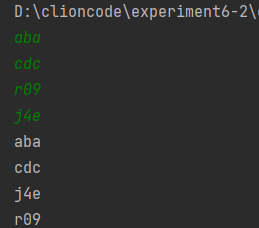


图6-5 源程序完善与修改替换题1测试用例1运行结果截图

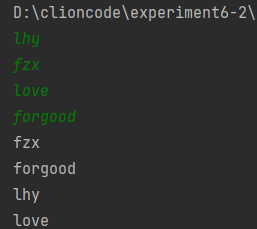


图6-6 源程序完善与修改替换题1测试用例2运行结果截图

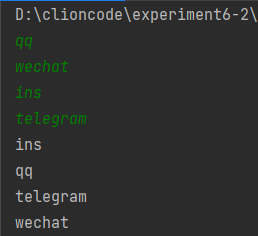


图6-7 源程序完善与修改替换题1测试用例3运行结果截图

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

（2）下面源程序通过函数指针和菜单选择来调用库函数实现字符串操作；串复制strcpy、串连接strcat或串分解strtok。

①请在源程序中的下划线处填写合适的代码来完善该程序，使之能按照要求输出下面结果：

1 copy string.

2 connect string.

3 parse string.

4 exit.

input a number (1-4) please!

2↙ （键盘输入）

input the first string please!

the more you learn,↙ （键盘输入）

input the second string please!

the more you get. ↙ （键盘输入）

the result is the more you learn, the more you get.

# include<stdio.h>

# include<string.h>

int main (void)

{

void \*p\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

char a[80], b[80], \*result;

int choice;

while(1)

{

do

{

printf("\t\t1 copy string.\n");

printf("\t\t2 connect string.\n");

printf("\t\t3 parse string.\n");

printf("\t\t4 exit.\n");

printf("\t\tinput a number (1-4) please.\n");

scanf("%d", &choice);

}while(choice<1 || choice>4);

switch(choice)

{

case 1: p = strcpy; break;

case 2: p = strcat; break;

case 3: p = strok; break;

case 4: goto down;

}

getchar();

printf("input the first string please!\n");

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

printf("input the second string please!\n");

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

result = \_\_\_\_\_\_\_\_p\_\_\_(a, b);

printf("the result is %s\n", result);

}

down:

return 0;

}

**解答：**

1. 填空：
2. char\* (\*p)(char\*,const char\*)；
3. gets（a）；
4. gets（b）；
5. P
6. 填空后的源程序清单如下：、

# include<stdio.h>

# include<string.h>

int main (void)

{

char\* (\*p)(char\*,const char\*);

char a[80], b[80], \*result;

int choice;

while(1)

{

do

{

printf("\t\t1 copy string.\n");

printf("\t\t2 connect string.\n");

printf("\t\t3 parse string.\n");

printf("\t\t4 exit.\n");

printf("\t\tinput a number (1-4) please.\n");

scanf("%d", &choice);

}while(choice<1 || choice>4);

switch(choice)

{

case 1: p = strcpy; break;

case 2: p = strcat; break;

case 3: p = strtok; break;

case 4: goto down;

}

getchar();

printf("input the first string please!\n");

gets（a）；

printf("input the second string please!\n");

gets（b）；

result = p(a, b);

printf("the result is %s\n", result);

}

down:

return 0;

}

1. 修改后的运行结果如图6-8所示：

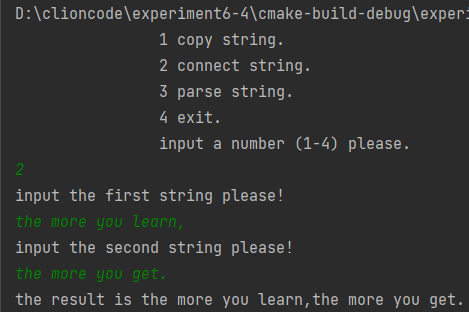


图6-8 源程序完善与修改替换题（2）运行结果截图

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

②函数指针的一个用途是用户散转程序，即通过一个转移表（函数指针数组）来实现多分枝函数处理，从而省去了大量的if语句或者switch语句。转移表中存放了各个函数的入口地址（函数名），根据条件的设定来查表选择执行相应的函数。请使用转移表而不是switch语句重写以上程序。

**解答：**

1. 解题思路：

创建一个函数指针，存放各个函数的地址，然后通过间访的方式调用函数。

1. 源程序清单如下：

# include<stdio.h>

# include<string.h>

int main (void)

{

char\* (\*p[4])(char\*,const char\*);

char a[80], b[80], \*result;

int choice;

while(1)

{

do

{

printf("\t\t1 copy string.\n");

printf("\t\t2 connect string.\n");

printf("\t\t3 parse string.\n");

printf("\t\t4 exit.\n");

printf("\t\tinput a number (1-4) please.\n");

scanf("%d", &choice);

}while(choice<1 || choice>4);

p[0]=strcpy;

p[1]=strcat;

p[2]=strtok;

if(choice==4) goto down;

getchar（）；

printf("input the first string please!\n");

gets（a）；

printf("input the second string please!\n");

gets（b）；

result = p[choice-1](a, b);

printf("the result is %s\n", result);

}

down:

return 0;

}

1. 修改后运行结果如图6-9所示：

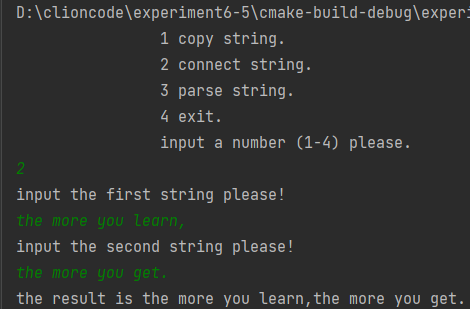


图6-9 源程序完善与修改替换题（2）运行结果截图

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

**3、跟踪调试题**

请按下面的要求对源程序进行操作，并回答问题和排除错误。

（1）单步执行。进入strcpy时watch窗口中s为何值？返回main时, watch窗口中s为何值？

（2）排除错误，使程序输出结果为：there is a boat on the lake.

#include "stdio.h"

char \*strcpy(char \*,char \*);

void main(void)

{

char a[20],b[60]="there is a boat on the lake.";

printf("%s\n",strcpy(a,b));

}

char \*strcpy(char \*s,char \*t)

{

while(\*s++=\*t++)

;

return (s);

}

**解答：**

1. 进入strcpy时watch窗口中s为\b, 返回main时, watch窗口中s为“ ”。
2. 修改strcpy函数，修改后源程序如下：

#include "stdio.h"

char \*strcpy(char \*,char \*);

void main(void)

{

char a[60],b[60]="there is a boat on the lake.";

printf("%s\n",strcpy(a,b));

}

char \*strcpy(char \*s,char \*t)

{

int i;

for(i=0;\*(t+i)!='\0';i++)

\*(s+i)=\*(t+i);

\*(s+i)='\0';

return (s);

}

1. 运行结果如图6-10所示：

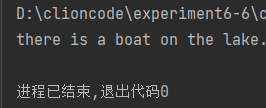


图6-10 跟踪调试题运行结果截图

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

**4.编程设计题**

（1）指定main函数的参数

在IDE（比如DevC++）中，选择“**运行**”｜“**参数**”菜单，在 “**传递给主程序的参数**”文本框中输入main函数的参数arg1 arg2 arg3，只输入命令行中文件名后的参数，文件名不作为参数输入，参数间以空格隔开。编写程序在命令行输出这三个参数。（注意不同IDE输入参数的方式不相同，可参考各个IDE的使用手册。）

解答：

1. 思路：

用for循环将输入并存储在argv数组中的参数打印出来。

1. 源程序清单如下：

#include <stdio.h>

int main(int argc,char\* argv[]) {

for (int i = 0; i < argc; ++i) {

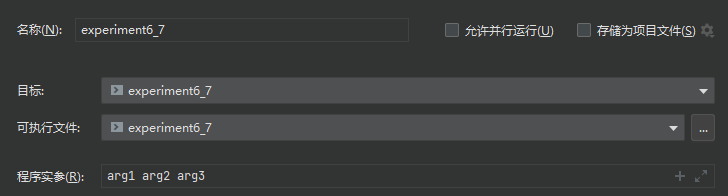
printf("%s\n",argv[i]);

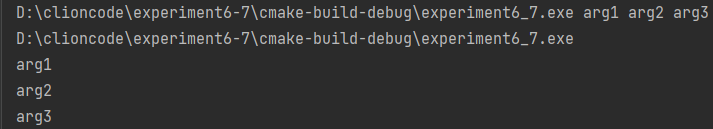
}

return 0;

}

1. 设置结果以及运行结果如图6-11,6-12所示：

图6-11 程序设计题（1）设置

图6-12 程序设计题（1）运行结果截图

（2） 一个长整型变量占4个字节，其中每个字节又分成高4位和低4位。输入一个长整型变量，要求从高字节开始，依次取出每个字节的高4位和低4位并以十六进制数字字符的形式进行显示，通过指针取出每字节。

**解答：**

（1）解题思路：

将输入的数字x强制转换为char型，并定义一个char型指针指向x的第4位，定义两个char型变量lowhalf，uphalf，lowhalf用于低4位字节，uphalf用于存储高4位字节。打印出来后，令p自减，重复4次即可得到结果。

（2）源程序清单如下：

#include <stdio.h>

int main()

{

long x;

scanf("%ld",&x);

char \*p=((char\*)&x+3);

char up\_half,low\_half;

for(int k=0;k<4;k++)

{

low\_half=(\*p)&0x0f;

up\_half=(\*p>>4)&0x0f;

printf("%llX",up\_half);

printf("%llX",low\_half);

p--;

}

return 0;

}

（3）测试：

1. 测试数据如表6-3所示。

表6-3 程序设计题（2）测试数据表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 实际结果 |
| 用例1 | 15 | 0000000F | 0000000F |
| 用例2 | 2147483647 | 7FFFFFFF | 7FFFFFFF |
| 用例3 | 0 | 00000000 | 00000000 |

b）程序运行结果如图6-13，图6-14，图6-15所示。

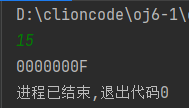


图6-13 程序设计题（2）测试用例1运行结果截图

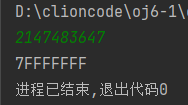


图6-14 程序设计题（2）测试用例2运行结果截图

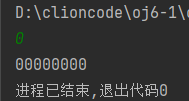


图6-15 程序设计题（2）测试用例3运行结果截图

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

（3） 旋转是图像处理的基本操作，编程实现一个将一个图像逆时针旋转90°。

提示：计算机中的图像可以用一个矩阵来表示，旋转一个图像就是旋转对应的矩阵。将旋转矩阵的功能定义成函数，通过使用指向数组元素的指针作为参数使该函数能处理任意大小的矩阵。要求在main函数中输入图像矩阵的行数n和列数m，接下来的n行每行输入m个整数，表示输入的图像。输出原始矩阵逆时针旋转90°后的矩阵。

**解答：**

（1）解题思路：

首先定义一个指针p指向第一行最后一个数并打印，再令p+m，指向第二行最后一个数并打印，直到最后一个数被打印并换行，此时令p指向第一行最后一个数，并令p自减，打印该数，每打印n个数换一次行，直至打印m行。

（2）源程序清单如下：

#include <stdio.h>

void turn(int \*p,int x,int y,int a[x][y]){

for(int i=0;i<x;i++)

{

if(i==x-1) printf("%d",a[i][y-1]);

else printf("%d ",a[i][y-1]);

}

printf("\n");

for(int j=0;j<y-1;j++)

{

for(int t=0;t<x;t++)

{

p=&a[t][y-1-j];

--p;

if(t==x-1)printf("%d",\*p);

else printf("%d ",\*p);

}

printf("\n");

}

}

int main() {

int x,y;

scanf("%d%d",&x,&y);

int a[x][y];

for(int i=0;i<x;i++)

for(int j=0;j<y;j++)

scanf(" %d",&a[i][j]);

int \*p=a;

turn(p,x,y,a);

return 0;

}

（3）测试：

a）测试数据如表6-5所示。

表6-5 程序设计题（4）测试数据表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 实际结果 |
| 用例1 | 2 3  153324 | 3 4  5 2  1 3 | 3 4  5 2  1 3 |
| 用例2 | 1. 1   13 | 13 | 13 |
| 用例3 | 5 5  12345  54321  12345  54321  12345 | 51515  42424  33333  24242  15151 | 51515  42424  33333  24242  15151 |

b）程序运行结果如图6-16，图6-17，图6-18所示。

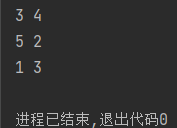


图6-16 程序设计题（3）测试用例1运行结果截图

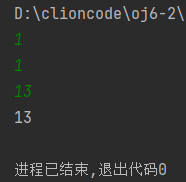


图6-17 程序设计题（3）测试用例2 运行结果截图

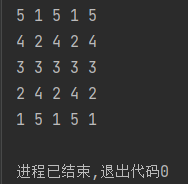


图6-18 程序设计题（3）测试用例3运行结果截图

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

（4） 输入n行文本，每行不超过80个字符，用字符指针数组指向键盘输入的n行文本（n不作为输入，可理解为循环输入多行，以CTRL+Z结束），删除每一行中的前置空格（' '）和水平制表符（'\t'）。要求：将删除一行文本中前置空格和水平制表符的功能定义成函数，在main函数中输出删除前置空格符的各行。（并在最后输出换行符。）

**解答：**

（1）解题思路：

定义一个delete函数，定义n=0，用于记录并删除前置空格和制表符，利用char型指针一个一个进行判断，若为前置空格或制表符则删除，并令n++，最后返回n。在主函数中，用for循环，打印数组中的前n项。

（2）源程序清单如下：

#include <stdio.h>

#include <stdio.h>

int delete(char \*p1,char \*p2,int i){

int n = 0,j = 0;

int flag = 0;

for(;j < i;j++){

if (((\*p1) !=' ')&&((\*p1) != '\t')){

\*(p2++) = \*(p1++);

n++;

flag = 1;

if ((\*(p1-1)) == '\n') flag = 0;

}

else if((\*p1)=='\t') p1++;

else if(((\*p1)==' ')&&(flag == 0)) p1++;

else {

\*(p2++) = \*(p1++);

n++;

}

}

return n;

}

int main(){

char a[99],b[99];

int i = 0;

while ((scanf("%c",&a[i])) != EOF) i++;

char \*p1 = &a[0];

char \*p2 = &b[0];

int n = delete(p1,p2,--i);

for (i = 0;i < n;i++) printf("%c",b[i]);

printf("\n");

return 0;

}

（3）测试：

1. 测试数据如表6-6所示。

表6-6 程序设计题（4）测试数据表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 实际结果 |
| 用例1 | 424\t 24  Ctrl+z | 42424 | 42424 |
| 用例2 | 123\t45\ta  Ctrl+z | 12345a | 12345a |
| 用例3 | \tfjsl \t kl  Slkfjs  Ctrl+z | fjslkl  Slkfjs | fjslkl  Slkfjs |

1. 程序运行结果如图6-19，图6-20，图6-21所示。

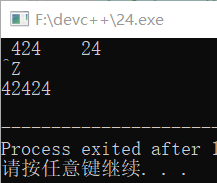


图6-19 程序设计题（4）测试用例1运行结果截图

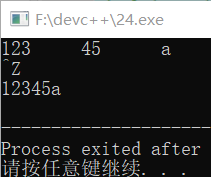


图6-20 程序设计题（4）测试用例2 运行结果截图

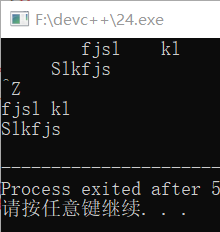


图6-21 程序设计题（4）测试用例3 运行结果截图

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

（5）编写8个任务函数，一个scheduler调度函数和一个execute执行函数。仅在main函数中调用scheduler函数，scheduler函数要求用最快的方式调度执行用户指定的任务函数。①先设计task0, task1, task2, task3, task4, task5, task6, task7共8个任务函数，每个任务函数的任务就是输出该任务被调用的字符串。例如，第0个任务函数输出“task0 is called!”，第1个任务函数输出“task1 is called!”，以此类推。②scheduler函数根据键盘输入的数字字符的先后顺序，一次调度选择对应的任务函数。例如，输入：1350并回车，则scheduler函数一次调度选择task1, task3, taks5, task0，然后以函数指针数组和任务个数为参数将调度选择结果传递给execute函数并调用execute函数。③execute函数根据scheduler函数传递的指针数组和任务个数为参数，按照指定的先后顺序依此调用执行选定的任务函数。

当输入13607122并回车，程序运行结果应当如下：

task1 is called!

task3 is called!

task6 is called!

task0 is called!

task7 is called!

task1 is called!

task2 is called!

task2 is called!

**解答：**

（1）解题思路：

定义7个函数，分别用来输出语句，定义char型指针，一个一个读取输入的数字并存入数组中，再定义一个指针数组，指向每一个输出函数，在execute函数中按位调用指针数组中的函数。

（2）源程序清单如下：

#include <stdio.h>

void execute(void(\*p[])(),int n);

int task0(){

printf("task0 is called!\n");

}

void task1(){

printf("task1 is called!\n");

}

void task2(){

printf("task2 is called!\n");

}

void task3(){

printf("task3 is called!\n");

}

void task4(){

printf("task4 is called!\n");

}

void task5(){

printf("task5 is called!\n");

}

void task6(){

printf("task6 is called!\n");

}

void task7(){

printf("task7 is called!\n");

}

void scheduler(){

int count=0;

char b[10];

scanf("%s",b);

for(int i=0;b[i]!='\0';count++)

i++;

int \*p[count];

for(int j=0;j<count;j++)

switch (b[j]-'0') {

case 0:p[j]= (int \*) task0;break;

case 1:p[j]= (int \*) task1;break;

case 2:p[j]= (int \*) task2;break;

case 3:p[j]= (int \*) task3;break;

case 4:p[j]= (int \*) task4;break;

case 5:p[j]= (int \*) task5;break;

case 6:p[j]= (int \*) task6;break;

case 7:p[j]= (int \*) task7;break;

}

execute(p, count);

}

void execute(void(\*p[])(),int n){

for(int i=0;i<n;i++)

p[i]();

}

int main() {

scheduler();

return 0;

}

（3）测试：

1. 测试数据如表6-7所示。

表6-7 程序设计题（5）测试数据表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 实际结果 |
| 用例1 | 123456 | task1 is called!  task2 is called!  task3 is called!  task4 is called!  task5 is called!  task6 is called! | task1 is called!  task2 is called!  task3 is called!  task4 is called!  task5 is called!  task6 is called! |
| 用例2 | 65473 | task6 is called!  task5 is called!  task4 is called!  task7 is called!  task3 is called! | task6 is called!  task5 is called!  task4 is called!  task7 is called!  task3 is called! |
| 用例3 | 0 | task0 is called! | task0 is called! |

1. 程序运行结果如图6-22，图6-23，图6-24所示。

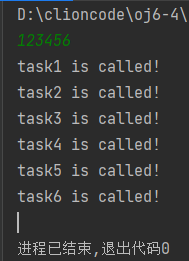


图6-22 程序设计题（5）测试用例1运行结果截图

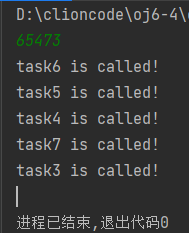


图6-23 程序设计题（5）测试用例2 运行结果截图

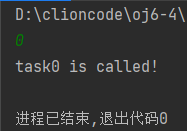


图6-24 程序设计题（5）测试用例3运行结果截图

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

## 6.3实验小结

在本次实验中，主要问题就是出在函数指针和指针函数那块，使用的时候会犯迷糊，容易搞混淆，这块还需要加强。

# 实验7 结构与联合实验

## 7.1实验目的

1．通过实验，熟悉和掌握结构的说明和引用、结构的指针、结构数组、以及函数中使用结构的方法。

2．通过实验，掌握动态储存分配函数的用法，掌握自引用结构，单向链表的创建、遍历、结点的增删、查找等操作。

3．了解字段结构和联合的用法。

## 7.2实验题目及要求

**1．表达式求值的程序验证题**

设有说明：

char u[]="UVWXYZ";

char v[]="xyz";

struct T{

int x;

char c;

char \*t;

}a[]={{11,ˊAˊ,u},{100, ˊBˊ,v}},\*p=a;

请先自己计算下面表达式的值，然后通过编程计算来加以验证。(各表达式相互无关)

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **表达式** | **计算值** | **验证值** |
| 1 | (++p)->x | 100 | 100 |
| 2 | p++,p->c | ‘B’ | ‘B’ |
| 3 | \*p++->t,\*p->t | ‘x’ | ‘x’ |
| 4 | \*(++p)->t | ‘x’ | ‘x’ |
| 5 | \*++p->t | ‘V’ | ‘V’ |
| 6 | ++\*p->t | ‘V’ | ‘V’ |

**解答：**

各表达式输出结果如下图所示：

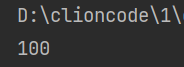


图7-1 程序验证题（1）的运行结果



图7-2 程序验证题（2）的运行结果

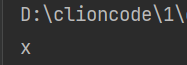


图7-3 程序验证题（3）的运行结果

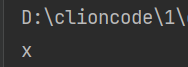


图7-4 程序验证题（4）的运行结果

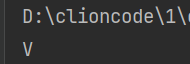


图7-5 程序验证题（5）的运行结果



图7-6 程序验证题（6）的运行结果

**2．源程序修改替换题**

给定一批整数，以0作为结束标志且不作为结点，将其建成一个先进先出的链表，先进先出链表的指头指针始终指向最先创建的结点（链头），先建结点指向后建结点，后建结点始终是尾结点。

1. 源程序中存在什么样的错误（先观察执行结果）？对程序进行修改、调试，使之能够正确完成指定任务。

源程序如下：

1 #include "stdio.h"

2 #include "stdlib.h"

3 struct s\_list{

4 int data; /\* 数据域 \*/

5 struct s\_list \*next; /\* 指针域 \*/

6 } ;

7 void create\_list (struct s\_list \*headp,int \*p);

8 void main(void)

9 {

10 struct s\_list \*head=NULL,\*p;

11 int s[]={1,2,3,4,5,6,7,8,0}; /\* 0为结束标记 \*/

12 create\_list(head,s); /\* 创建新链表 \*/

13 p=head; /\*遍历指针p指向链头 \*/

14 while(p){

15 printf("%d\t",p->data); /\* 输出数据域的值 \*/

16 p=p->next; /\*遍历指针p指向下一结点 \*/

17 }

18 printf("\n");

19 }

20 void create\_list(struct s\_list \*headp,int \*p)

21 {

22 struct s\_list \* loc\_head=NULL,\*tail;

23 if(p[0]==0) /\* 相当于\*p==0 \*/

24 ;

25 else { /\* loc\_head指向动态分配的第一个结点 \*/

26 loc\_head=(struct s\_list \*)malloc(sizeof(struct s\_list));

27 loc\_head->data=\*p++; /\* 对数据域赋值 \*/

28 tail=loc\_head; /\* tail指向第一个结点 \*/

29 while(\*p){ /\* tail所指结点的指针域指向动态创建的结点 \*/

30 tail->next=(struct s\_list \*)malloc(sizeof(struct s\_list));

31 tail=tail->next; /\* tail指向新创建的结点 \*/

32 tail->data=\*p++; /\* 向新创建的结点的数据域赋值 \*/

33 }

34 tail->next=NULL; /\* 对指针域赋NULL值 \*/

35 }

36 headp=loc\_head; /\* 使头指针headp指向新创建的链表 \*/

37 }

**解答：**

1. 错误修改：
2. 第7行，第20行中在create\_list函数中，由于headp是一个变化的指针，故应将参数中的\*headp改为双重指针\*\*headp。
3. 第12行，由于传入的参数是双重指针，故将create\_list(head,s);改为create\_list(&head,s);
4. 第19行，应释放空间，增添语句

p=head;

for (int i = 0; i < 8; ++i) {

free(p),p=p->next;

}

1. 第36行，将headp改为\*headp。
2. 修改后的源程序清单如下：

#include "stdio.h"

#include "stdlib.h"

struct s\_list{

int data; /\* 数据域 \*/

struct s\_list \*next; /\* 指针域 \*/

} ;

void create\_list (struct s\_list \*\*headp,int \*p);

void main(void)

{

struct s\_list \*head=NULL,\*p;

int s[]={1,2,3,4,5,6,7,8,0}; /\* 0为结束标记 \*/

create\_list(&head,s); /\* 创建新链表 \*/

p=head; /\*遍历指针p指向链头 \*/

while(p){

printf("%d\t",p->data); /\* 输出数据域的值 \*/

p=p->next; /\*遍历指针p指向下一结点 \*/

}

printf("\n");

p=head;

for (int i = 0; i < 8; ++i) {

free(p),p=p->next;

}

}

void create\_list(struct s\_list \*\*headp,int \*p)

{

int i=0;

struct s\_list \* loc\_head=NULL,\*tail;

if(p[0]==0) /\* 相当于\*p==0 \*/;

else { /\* loc\_head指向动态分配的第一个结点 \*/

loc\_head=(struct s\_list \*)malloc(sizeof(struct s\_list));

loc\_head->data=\*(p+(i++)); /\* 对数据域赋值 \*/

tail=loc\_head; /\* tail指向第一个结点 \*/

while(\*(p+i)){ /\* tail所指结点的指针域指向动态创建的结点 \*/

tail->next=(struct s\_list \*)malloc(sizeof(struct s\_list));

tail=tail->next; /\* tail指向新创建的结点 \*/

tail->data=\*(p+(i++)); /\* 向新创建的结点的数据域赋值 \*/

}

tail->next=NULL; /\* 对指针域赋NULL值 \*/

}

\*headp=loc\_head; /\* 使头指针headp指向新创建的链表 \*/

}

1. 修改后的运行结果如图7-7所示。

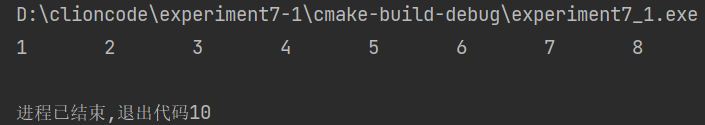


图7-7 程序改错题的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

1. 修改替换create\_list函数，将其建成一个后进先出的链表，后进先出链表的头指针始终指向最后创建的结点（链头），后建结点指向先建结点，先建结点始终是尾结点。

**解答：**

1. 修改思路：

修改create\_list函数，定义两个结构体指针head1，head2用于创建链表，while循环中head1用于申请空间并输入数值，并将head1的next指针指向head2，最后将head2指向head1，其余思路与1题一致。

1. 修改后的源程序清单如下：

#include "stdio.h"

#include "stdlib.h"

struct s\_list{

int data; /\* 数据域 \*/

struct s\_list \*next; /\* 指针域 \*/

} ;

void create\_list (struct s\_list \*\*headp,int \*p);

void main(void)

{

struct s\_list \*head=NULL,\*p;

int s[]={1,2,3,4,5,6,7,8,0}; /\* 0为结束标记 \*/

create\_list(&head,s); /\* 创建新链表 \*/

p=head; /\*遍历指针p指向链头 \*/

while(p){

printf("%d\t",p->data); /\* 输出数据域的值 \*/

p=p->next; /\*遍历指针p指向下一结点 \*/

}

p=head;

for (int i = 0; i < 8; ++i) {

free(p),p=p->next;

}

printf("\n");

}

void create\_list(struct s\_list \*\*headp,int \*p)

{

int i=0;

struct s\_list \* loc\_tail=NULL,\*head1,\*head2;

if(p[0]==0) /\* 相当于\*p==0 \*/;

else { /\* loc\_tail指向动态分配的最后一个结点 \*/

loc\_tail=(struct s\_list \*)malloc(sizeof(struct s\_list));

loc\_tail->data=\*(p+(i++)); /\* 对数据域赋值 \*/

head2=loc\_tail; /\* tail指向最后一个结点 \*/

while(\*(p+i)){ /\* head2所指结点的指针域指向动态创建的结点 \*/

head1=(struct s\_list\*)malloc(sizeof (struct s\_list));

head1->next=head2;

head1->data=\*(p+(i++));

head2=head1;

}

loc\_tail->next=NULL; /\* 对指针域赋NULL值 \*/

}

\*headp=head2; /\* 使头指针headp指向新创建的链表 \*/

}

1. 修改后的运行结果如图7-8所示：

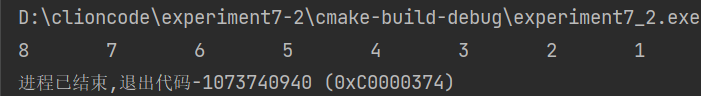


图7-8 程序修改题的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

**3．程序设计题**

1）用单向链表建立一张班级成绩单，包括每个学生的学号、姓名、英语、高等数学、普通物理、C语言程序设计四门课程的成绩。用菜单实现下列功能： ① 输入每个学生的各项信息。 ② 输出每个学生的各项信息。 ③ 修改指定学生的指定数据项的内容。 ④ 统计每个同学的平均成绩（保留2位小数）。 ⑤ 输出各位同学的学号、姓名、四门课程的总成绩和平均成绩。⑥增加按照平均成绩进行升序排序的函数，写出用交换结点数据域的方法升序排序的函数，排序可用选择法或冒泡法。进一步写出用交换结点指针域的方法升序排序的函数。

**解答：**

1. 解题思路：
2. 定义结构体student，包含学号，成绩等数据，并定义静态全局变量n，aver。
3. 定义input函数，scanf用于接收键盘输入的数字n，再用for循环n次用于接收输入结构体的数据，同时申请空间，创建链表。
4. 定义output函数，利用while循环，打印数据，直到最后一个结点。
5. 定义change函数，用于修改数据，以头指针以及学号为参数，先利用while循环找到输入的学号所在的结点，然后根据键盘上输入的数字更改相应的数据。
6. 定义average函数，用于求平均值，当循环变量小于n时，进行求值，并每次求完值后将结点向后移一位。
7. 定义output1函数，用于输出总分数以及平均值，当循环变量小于n时，求总分数并打印学号，总分，均值。
8. 定义sort函数，用于根据均分排序，采用冒泡排序法和更换数据域的方式，根据student.aver判断是否进行交换位置，并采用strcpy更换姓名。
9. 主函数中将6个函数利用switch串接，当输入为0时，执行return0，即结束程序。

1. 源程序代码清单如下：

#include <stdio.h>

#include "stdlib.h"

#include "string.h"

static int n;

struct student{

int num;

char name[20];

int english,math,physics,c;

float aver;

struct student \*next;

};

void input(struct student\*p){

scanf("%d",&n);

for(int i=0;i<n;i++)

{

struct student\*s=(struct student\*) malloc(sizeof (struct student));

scanf("%d%s%d%d%d%d",&s->num,s->name,&s->english,&s->math,&s->physics,&s->c);

p->next=s;

s->next=NULL;

p=s;

}

printf("完成了%d位同学的成绩输入。\n",n);

}

void output(struct student\*p){

while(p->next!=NULL)

{

printf("%d %s %d %d %d %d\n",p->next->num,p->next->name,p->next->english,p->next->math,p->next->physics,p->next->c);

p=p->next;

}

}

int change(struct student\*p,int a){

int meun2;

while(p->next->num!=a)

p=p->next;

if(p->next==NULL) return 0;

else

{

scanf("%d",&meun2);

switch (meun2) {

case 0:

scanf("%s",p->name);break;

case 1:

scanf("%d",&p->english);break;

case 2:

scanf("%d",&p->math);break;

case 3:

scanf("%d",&p->physics);break;

case 4:

scanf("%d",&p->c);break;

}

return 1;

}

}

void average(struct student\*p){

for(int i=0;i<n;i++)

{ p->next->aver=((float )p->next->c+(float )p->next->physics+(float)p->next->math+(float)p->next->english)/4;

printf("%d %s %2.2f\n",p->next->num,p->next->name,p->next->aver);

p=p->next;

}

}

void output1(struct student\*p){

int total;

for(int i=0;i<n;i++)

{

total=p->next->english+p->next->math+p->next->c+p->next->physics; printf("%d %s %d %2.2f\n",p->next->num,p->next->name,total,p->next->aver);

p=p->next;

}

}

void sort(struct student\*p1){

float temp;

struct student \*temp2=p1;

struct student \*p2;

int t,i,j;

char a[20];

for(int i=0;i<n;i++)

{

p1->next->aver=((float )p1->next->c+(float )p1->next->physics+(float)p1->next->math+(float)p1->next->english)/4;

p1=p1->next;

}

p1=temp2;

for(i=0,p1=p1->next;i<n-1;i++,p1=p1->next)

for(j=0,p2=p1->next;j<n-i-1;j++,p2=p2->next)

{

if(p1->aver>p2->aver)

{

t=p1->num;

p1->num=p2->num;

p2->num=t;

temp=p1->aver;

p1->aver=p2->aver;

p2->aver=temp;

strcpy(a,p1->name);

strcpy(p1->name,p2->name);

strcpy(p2->name,a);

}

}

for(int i=0;i<n;i++,temp2=temp2->next)

printf("%d %s %2.2f\n",temp2->next->num,temp2->next->name,temp2->next->aver);

}

int main() {

struct student\*head=(struct student\*)malloc(sizeof (struct student));

head->next=NULL;

struct student\*p;

p=head;

int meun1=0,number;

while((scanf("%d",&meun1))!=EOF)

{

switch (meun1) {

case 0: return 0;

case 1:

input(p),p=head;break;

case 2:

output(p),p=head;break;

case 3:

scanf("%d",&number);

change(p,number),p=head;break;

case 4:

average(p),p=head;break;

case 5:

output1(p),p=head;break;

case 6:

sort(p),p=head;break;

}

}

return 0;

}

1. 测试：
2. 测试数据如表7-1所示：

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例1 | 1 5  2021001 Jack 90 92 87 95 2021002 Mike 85 70 75 90 2021003 Joe 77 86 90 75 2021004 Andy 95 97 92 95 2021005 Rose 90 87 88 89 | 完成了5位同学的成绩输入。 | 完成了5位同学的成绩输入。 |
| 用例2 | 1 5  2021001 Jack 90 92 87 95 2021002 Mike 85 70 75 90 2021003 Joe 77 86 90 75 2021004 Andy 95 97 92 95 2021005 Rose 90 87 88 89  4 | 完成了5位同学的成绩输入。  2021001 Jack 91.00  2021002 Mike 80.00  2021003 Joe 82.00  2021004 Andy 94.75  2021005 Rose 88.50 | 完成了5位同学的成绩输入。  2021001 Jack 91.00  2021002 Mike 80.00  2021003 Joe 82.00  2021004 Andy 94.75  2021005 Rose 88.50 |
| 用例3 | 1 5  2021001 Jack 90 92 87 95 2021002 Mike 85 70 75 90 2021003 Joe 77 86 90 75 2021004 Andy 95 97 92 95 2021005 Rose 90 87 88 89  5  0 | 完成了5位同学的成绩输入。  2021001 Jack 364 91.00  2021002 Mike 320 80.00  2021003 Joe 328 82.00  2021004 Andy 379 94.00  2021005 Rose 354 88.00  结束 | 完成了5位同学的成绩输入。  2021001 Jack 364 91.00  2021002 Mike 320 80.00  2021003 Joe 328 82.00  2021004 Andy 379 94.00  2021005 Rose 354 88.00  结束 |

表7-1 程序设计题（1）测试数据

b）运行结果如图7-9,7-10,7-11所示

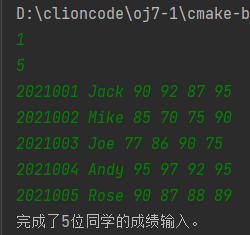


图7-9 程序设计题用例（1）的运行结果



图7-10 程序设计题（1）用例2的运行结果



图7-11 程序设计题（1）用例3的运行结果

2）将排序方法更改为更改结点域，重新定义sort函数。

解答：

1. 解题思路：

定义pre，current，after和temp4个结构体指针，仍然使用冒泡排序，从第二个节点开始，依次向后遍历，每次不论执不执行交换，都将pre，current，after指针向后移一位，如此反复知道遍历到链表尾部。

1. 源程序代码清单如下：

#include <stdio.h>

#include "stdlib.h"

static int n;

struct student{

int num;

char name[20];

int english,math,physics,c;

float aver;

struct student \*next;

};

void input(struct student\*p){

scanf("%d",&n);

for(int i=0;i<n;i++)

{

struct student\*s=(struct student\*) malloc(sizeof (struct student));

scanf("%d%s%d%d%d%d",&s->num,s->name,&s->english,&s->math,&s->physics,&s->c);

p->next=s;

s->next=NULL;

p=s;

}

printf("完成了%d位同学的成绩输入。\n",n);

}

void output(struct student\*p){

while(p->next!=NULL)

{

printf("%d %s %d %d %d %d\n",p->next->num,p->next->name,p->next->english,p->next->math,p->next->physics,p->next->c);

p=p->next;

}

}

int change(struct student\*p,int a){

int meun2;

while(p->next->num!=a)

p=p->next;

if(p->next==NULL) return 0;

else

{

scanf("%d",&meun2);

switch (meun2) {

case 0:

scanf("%s",p->name);break;

case 1:

scanf("%d",&p->english);break;

case 2:

scanf("%d",&p->math);break;

case 3:

scanf("%d",&p->physics);break;

case 4:

scanf("%d",&p->c);break;

}

return 1;

}

}

void average(struct student\*p){

for(int i=0;i<n;i++)

{

p->next->aver=((float )p->next->c+(float )p->next->physics+(float)p->next->math+(float)p->next->english)/4;

printf("%d %s %2.2f\n",p->next->num,p->next->name,p->next->aver);

p=p->next;

}

}

void output1(struct student\*p){

int total;

for(int i=0;i<n;i++)

{

total=p->next->english+p->next->math+p->next->c+p->next->physics;

printf("%d %s %d %2.2f\n",p->next->num,p->next->name,total,p->next->aver);

p=p->next;

}

}

void sort(struct student\*head){

head=head->next;

struct student\*pre,\*current,\*after,\*temp,\*p;

p=head;

int i,j;

for(i=0;i<n;i++)

{

head->aver=((float )head->c+(float )head->physics+(float)head->math+(float)head->english)/4;

head=head->next;

}

head=p;

for(i=0 ;i<n-1; i++) {

pre = head;

current = pre->next;

if (pre->aver > current->aver) {

temp = current->next;

current->next = pre;

pre->next = temp;

head = current;

}

for (j = 0, pre = head; j<n-i-2; j++, pre = pre->next) {

current = pre->next;

after = current->next;

if (current->aver >after->aver) {

pre->next = after;

current->next = after->next;

after->next = current;

}

}

}

for(i=0;i<n;i++,head=head->next)

printf("%d %s %2.2f\n",head->num,head->name,head->aver);

}

int main() {

struct student\*head=(struct student\*)malloc(sizeof (struct student));

head->next=NULL;

struct student\*p;

p=head;

int meun1=0,number;

while((scanf("%d",&meun1))!=EOF)

{

switch (meun1) {

case 0: return 0;

case 1:

input(p),p=head;break;

case 2:

output(p),p=head;break;

case 3:

scanf("%d",&number);

change(p,number),p=head;break;

case 4:

average(p),p=head;break;

case 5:

output1(p),p=head;break;

case 6:

sort(p),p=head;break;

}

}

return 0;

}

1. 测试：
2. 测试数据如表7-2所示

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例1 | 1 5  2021001 Jack 90 92 87 95 2021002 Mike 85 70 75 90 2021003 Joe 77 86 90 75 2021004 Andy 95 97 92 95 2021005 Rose 90 87 88 89 | 完成了5位同学的成绩输入。 | 完成了5位同学的成绩输入。 |
| 用例2 | 1 5  2021001 Jack 90 92 87 95 2021002 Mike 85 70 75 90 2021003 Joe 77 86 90 75 2021004 Andy 95 97 92 95 2021005 Rose 90 87 88 89  4 | 完成了5位同学的成绩输入。  2021001 Jack 91.00  2021002 Mike 80.00  2021003 Joe 82.00  2021004 Andy 94.75  2021005 Rose 88.50 | 完成了5位同学的成绩输入。  2021001 Jack 91.00  2021002 Mike 80.00  2021003 Joe 82.00  2021004 Andy 94.75  2021005 Rose 88.50 |
| 用例3 | 1 5  2021001 Jack 90 92 87 95 2021002 Mike 85 70 75 90 2021003 Joe 77 86 90 75 2021004 Andy 95 97 92 95 2021005 Rose 90 87 88 89  6  0 | 完成了5位同学的成绩输入。  2021002 Mike 80.00  2021003 Joe 82.00  2021005 Rose 88.50  2021001 Jack 91.00  2021004 Andy 94.75  结束 | 完成了5位同学的成绩输入。  2021002 Mike 80.00  2021003 Joe 82.00  2021005 Rose 88.50  2021001 Jack 91.00  2021004 Andy 94.75  结束 |

表7-2 程序设计题（2）测试数据

b）运行结果如图7-12,7-13,7-14所示

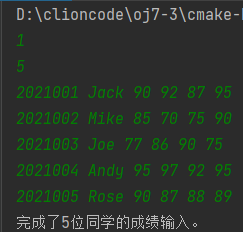


图7-12 程序设计题（2）用例1的运行结果



图7-13 程序设计题（2）用例2的运行结果



图7-14 程序设计题（2）用例3的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

## 7.3实验小结

本次实验我个人认为最难的点在于链表交换指针域的执行，我在实验过程中也借助了画图等手段试图去理解它，但是仍未掌握的很透彻，还需要加强这方面的练习。

# 实验8 文件操作实验

## 8.1 实验目的

1. 熟悉文本文件和二进制文件在磁盘中的存储方式；
2. 熟练掌握流式文件的读写方法。

## 8.2 实验内容

**8.2.1 文件类型的程序验证题**

设有程序：

1. #include <stdio.h>
2. int main(void)
3. {
4. short a=0x253f,b=0x7b7d;
5. char ch;
6. FILE \*fp1,\*fp2;
7. fp1=fopen("d:\\abc1.bin","wb+");
8. fp2=fopen("d:\\abc2.txt","w+");
9. fwrite(&a,sizeof(short),1,fp1);
10. fwrite(&b,sizeof(short),1,fp1);
11. fprintf(fp2,"%hx %hx",a,b);
12. rewind(fp1); rewind(fp2);
13. while((ch = fgetc(fp1)) != EOF)
14. putchar(ch);
15. putchar('\n');
17. while((ch = fgetc(fp2)) != EOF)
18. putchar(ch);
19. putchar('\n');
20. fclose(fp1);
21. fclose(fp2);
22. return 0;
23. }
24. 请思考程序的输出结果，然后通过上机运行来加以验证。
25. 将两处sizeof(short)均改为sizeof(char)结果有什么不同，为什么？
26. 将fprintf(fp2,"%hx %hx",a,b) 改为 fprintf(fp2,"%d %d",a,b)结果有什么不同。

**解答：**

1. 猜测结果为：

?%}{

253f 7b7d

上机运行得到的结果如图8-1所示：

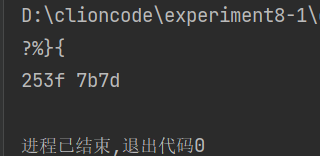


图8-1 验证题1运行结果截图

与猜测的结果一致；

1. sizeof(char)的结果是1，因此，fwrite函数从文件中每次读取的字节数为1；其中在内存视图中如图8-2所示；sizeof(char)运行的结果如图8-3所示：

图8-2 验证题1内存视图截图

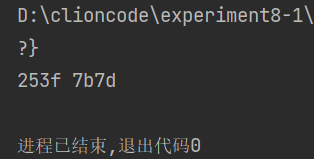


图8-3 验证题1修改后运行结果截图

1. 将fprintf(fp2,"%hx %hx",a,b) 改为 fprintf(fp2,"%d %d",a,b)后，运行结果如图8-4所示：

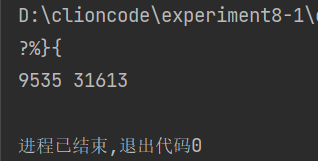


图8-4 验证题1修改后运行结果截图

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

**8.2.2 源程序修改替换**

将指定的文本文件内容在屏幕上显示出来，命令行的格式为：

type filename

1. 源程序中存在什么样的逻辑错误（先观察执行结果）？对程序进行修改、调试，使之能够正确完成指定任务。
2. #include<stdio.h>
3. #include<stdlib.h>
4. int main(int argc, char\* argv[])
5. {
6. char ch;
7. FILE \*fp;
8. if(argc!=2){
9. printf("Arguments error!\n");
10. exit(-1);
11. }
12. if((fp=fopen(argv[1],"r"))==NULL){ /\* fp 指向 filename \*/
13. printf("Can't open %s file!\n",argv[1]);
14. exit(-1);
15. }
16. while(ch=fgetc(fp)!=EOF) /\* 从filename中读字符 \*/
17. putchar(ch); /\* 向显示器中写字符 \*/
18. fclose(fp); /\* 关闭filename \*/
19. return 0;
20. }
21. 用输入输出重定向freopen改写main函数。

**解答：**

1. 程序改错：

a）第15行while(ch=fgetc(fp)!=EOF)中，！=的优先级高于=，因此需要在靠前的表达式外补充括号，正确形式如下：while((ch=fgetc(fp))!=EOF)

1. 源代码清单：
2. #include<stdio.h>
3. #include<stdlib.h>
4. int main(int argc, char\* argv[]) {
5. char ch;
6. FILE \*fp;
7. if(argc!=2) {
8. printf("Arguments error!\n");
9. exit(-1);
10. }
11. if((fp=fopen(argv[1],"r"))==NULL) { /\* fp 指向 filename \*/
12. printf("Can't open %s file!\n",argv[1]);
13. exit(-1);
14. }
15. while((ch=fgetc(fp))!=EOF) /\* 从filename中读字符 \*/ //
16. putchar(ch); /\* 向显示器中写字符 \*/
17. fclose(fp); /\* 关闭filename \*/
18. return 0;
19. }
20. 测试：

改错题1的形参输入如图8-5所示：

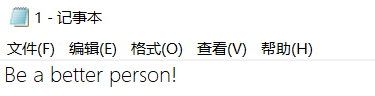


图8-5 源程序修改替换（1）输入形参内容截图

改错题（1）的运行结果如图8-6所示：

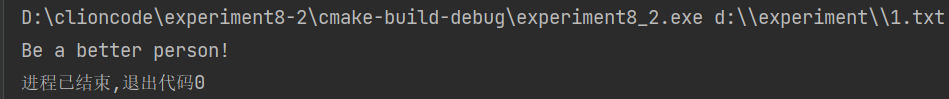


图8-6 源程序修改替换（1）修改后运行结果截图

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

1. 用freopen函数重写main函数：
2. #include<stdio.h>
3. #include<stdlib.h>
4. int main(int argc, char\* argv[]) {
5. char ch;
6. FILE \*fp;
7. if(argc!=2) {
8. printf("Arguments error!\n");
9. exit(-1);
10. }
11. if((fp=fopen(argv[1],"r"))==NULL) { /\* fp 指向 filename \*/
12. printf("Can't open %s file!\n",argv[1]);
13. exit(-1);
14. }
15. freopen(argv[1], "r", stdin);
16. while((ch=fgetc(fp))!=EOF) /\* 从filename中读字符 \*/
17. putchar(ch); /\* 向显示器中写字符 \*/
18. fclose(stdin); /\* 关闭filename \*/
19. return 0;
20. }
21. 测试：

改写main函数后运行结果如图8-7所示：

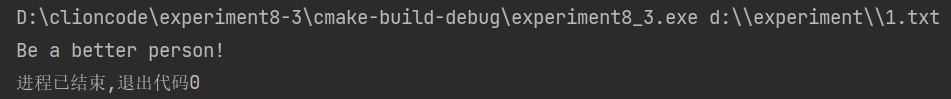


图8-7 源程序修改替换（2）修改后运行结果截图

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

**8.2.3 程序设计**

编写一个程序，用给定的字符串替换文件中的目标字符串，并显示输出替换的个数。注意：读取的文件路径请使用experiment/src/step8/source.txt

**解答：**

1. 解题思路如图8-8所示：

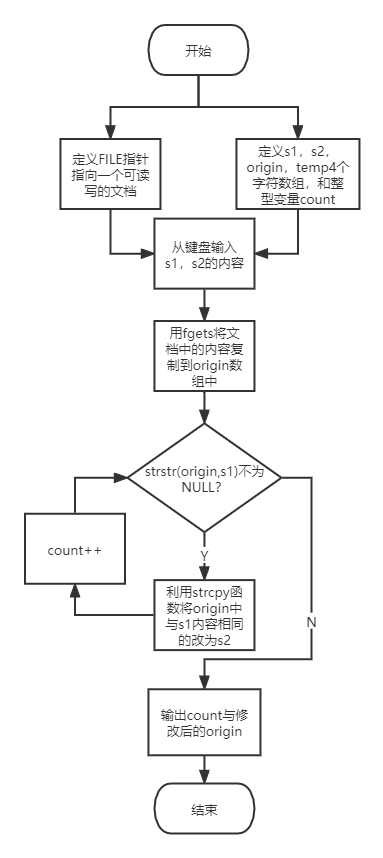


图8-8程序设计题流程图

1. 源程序代码清单如下：

#include <stdio.h>

#include "string.h"

int main() {

char s1[10],s2[10],origin[500]={'\0'},temp[500];

int count=0;

char \*p2;

FILE \*p= fopen("experiment/src/step8/source.txt","r+");

scanf("%s%s",s1,s2);

fgets(origin,500,p);

while((p2= strstr(origin,s1))){

strcpy(temp,p2+ strlen(s1));

strcpy(p2,s2);

strcpy(p2+ strlen(s2),temp);

count++;

}

printf("%d\n%s\n", count,origin);

fclose(p);

return 0;

}

1. 测试：（文档中的内容为There are moments in life when you miss someone so much that you just want to pick them from your dreams and hug them for real!）
2. 测试数据如表8-1所示：

表8-1程序设计题测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例1 | you they | 3 There are moments in life when they miss someone so much that they just want to pick them from theyr dreams and hug them for real! | 3  Therearemoments in life when they miss someone so much that they just want to pick them from theyr dreams and hug them for real! |
| 用例2 | are is | 1  There is moments in life when you miss someone so much that you just want to pick them from your dreams and hug them for  real! | 1  There is moments in life when you miss someone so much that you just want to pick them from your dreams and hug them for  real! |
| 用例3 | work w | 0  There are moments in life when you miss someone so much that you just want to pick them from your dreams and hug them fo  r real! | 0  There are moments in life when you miss someone so much that you just want to pick them from your dreams and hug them fo  r real! |

b）运行结果如图8-9,8-10,8-11所示

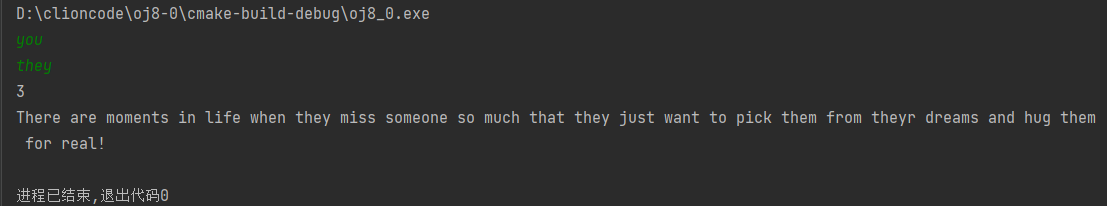


图8-9程序设计题测试用例（1）运行结果截图

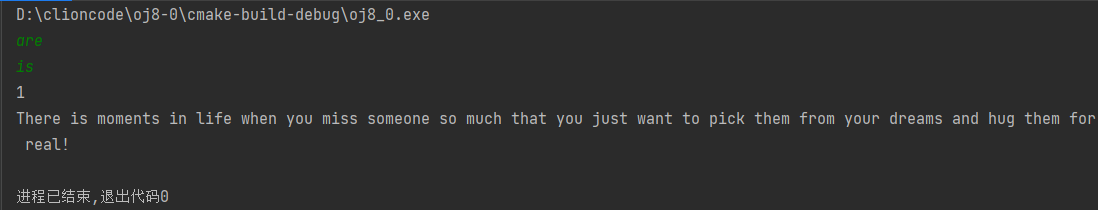


图8-10程序设计题测试用例（2）运行结果截图

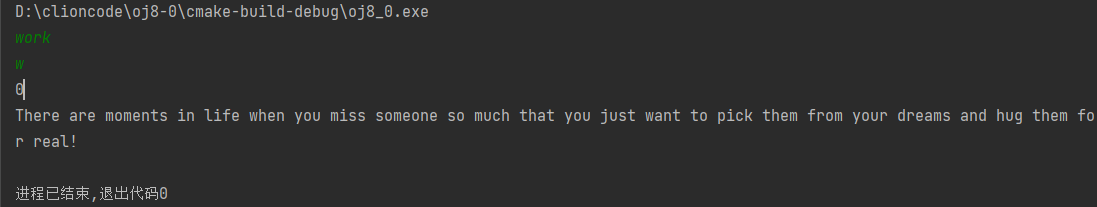


图8-11程序设计题测试用例（3）运行结果截图

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

## 8.3 实验小结

本次实验主要学习了C语言中流式文件的使用和文件的存储方式。让我认识到如何修改系统中的文件和文档，但对于函数的使用还有许多不熟之处，日后要多加练习。

## 小结：

本学期第一次接触c语言这类编程语言，对于一个之前从未接触过计算机语言的人来说，学习的过程中还是有不少困难的，尤其是对于一些难以理解的定义和算法，都花了我很多时间去理解，我在做头歌的时候也学到了一些方法去解决一些实际生活中的问题，作为一个计算机专业的学生，c语言是我计算机生涯的开始，也相当于一个路标，让我对于计算机的工作原理和程序的执行过程有了一些最基本的理解。我认为无论学习哪一种语言，对我们的专业都有很大的 ，但c语言无疑是最基础的，今后我们也将常常与他打交道，也要继续学习，争取做到炉火纯青。

参考文献

[1] 曹计昌,卢萍,李开. C语言程序设计, 北京：科学出版社, 2013

[2] 卢萍,李开,王多强等. C语言程序设计典型题解与实验指导, 北京：清华大学出版社, 2019